Especificación lógica TAD: LDEG

1. Elementos:

La estructura almacenará todo tipo de elementos, por lo que declararemos el tipo object en Java.

Será una estructura heterogénea, podemos almacenar todo tipo de datos primitivos.

1. Tipo de organización:

La lista enlazada tendrá una organización lineal, puesto que los nodos solo se conectan con el siguiente y con el anterior.

1. Dominio:

El dominio del TAD al ser heterogéneo, podemos almacenar cualquier tipo de dato.

1. Operaciones:
   1. Void insertar(Elemento):

* Nombre: insertar(E x)
* Descripción: inserta un elemento al principio de la lista.
* Datos de entrada: objeto.
* Salida: ninguno.
* Precondiciones: tener el objeto a agregar ya creado.
* Postcondiciones: tener el objeto agregado en la lista, como primer elemento.

* 1. Void insertar(elemento, indice):
* Nombre: insertar(E x, int e)
* Descripción: insertar un objeto en una posición concreta.
* Datos de entrada: un objeto o y un índice entero.
* Salida: ninguna.
* Precondiciones: ninguna.
* Postcondiciones: tener el objeto insertado en una posición concreta.
  1. Void insertarEnFin(elemento):
* Nombre: insertarEnFin(E x)
* Descripción: insertar un elemento al final de la lista.
* Datos de entrada: un objeto o.
* Salida: ninguna.
* Precondiciones: ninguna
* Postcondiciones: tener el objeto insertado al final de la lista.
  1. Int índiceDe(referencia):
* Nombre: indiceDe(E e)
* Descripción: pasamos una referencia, y si existe en la lista nos devuelve la posición, y si no existe, nos devuelve -1.
* Datos de entrada: Referencia de un objeto.
* Salida: Entero posición del objeto, o -1 si no existe.
* Precondiciones: ninguna.
* Postcondiciones: conocer la posición del objeto dentro de la lista.
  1. E recuperar(posición):
* Nombre: recuperar(int i).
* Descripción: devuelve el elemento que esté en la posición dada.
* Datos de entrada: Entero.
* Salida: objeto.
* Precondiciones: ninguna.
* Postcondiciones: tener el elemento de cierta posición.

// Este no está en los apuntes.

* 1. Boolean vaciarLista():
* Nombre: vaciarLista()
* Descripción: vacía la lista.
* Datos de entrada: ninguno.
* Salida: booleano.
* Precondiciones: ninguna.
* Postcondiciones: lista vacía.
  1. Boolean eliminar(referencia):
* Nombre: eliminar(E x)
* Descripción: elimina el nodo de un objeto que pasamos por referencia.
* Datos de entrada: objeto.
* Salida: booleano.
* Precondiciones: ninguna.
* Postcondiciones: elemento eliminado de la lista.

//este no está en los apuntes

* 1. E eliminar(posicion):
* Nombre: eliminar(int i)
* Descripción: eliminar el nodo que esté en la posición dada.
* Datos de entrada: Entero.
* Salida: objeto.
* Precondiciones: ninguna.
* Postcondiciones: eliminado el nodo de la posición dada.

//este no está en los apuntes

* 1. Boolean esVacia():
* Nombre: esVacia()
* Descripción: devuelve true si la lista está vacía, y false si la lista no está vacía.
* Datos de entrada: ninguno.
* Salida: booleano.
* Precondiciones: ninguno.
* Postcondiciones: saber si la lista está vacía.

//este no está en los apuntes

* 1. Boolean contiene(referencia):
* Nombre: contiene(object o)
* Descripción: devuelve true si la lista contiene el elemento, y false si no.
* Datos de entrada: objeto.
* Salida: booleano.
* Precondiciones: ninguna.
* Postcondiciones: saber si la lista contiene el elemento.

//este no está en lo apuntes

* 1. Boolean esIgual(referencia):
* Nombre: esIgual(object o)
* Descripción: nos devuelve true si el elemento es igual al dado, y false si no.
* Datos de entrada: objeto.
* Salida: booleano.
* Precondiciones: ninguna.
* Postcondiciones: saber si dos elementos son iguales.
  1. Int talla():
* Nombre: talla()
* Descripción: devuelve el número de elementos de la lista.
* Datos de entrada: ninguno.
* Salida: entero.
* Precondiciones: ninguna.
* Postcondiciones: saber el tamaño de la lista.

* 1. E[] toArray(array):
* Nombre: toArray(E[] a)
* Descripción:
* Datos de entrada:
* Salida:
* Precondiciones:
* Postcondiciones:
  1. String toString():
* Nombre: toString()
* Descripción:
* Datos de entrada:
* Salida:
* Precondiciones:
* Postcondiciones:

Notas:

<E extends Number> //para restringir el tipo de objetos con los que trabajamos; (enriquecer el TAD)