C:\Users\belen.moreno\Documents\NetBeansProjects\AplicaciónListaArray
\src\aplicacionlistaarray\Iterator.java

```
// Permite recorrer una estructura o colección de datos elemento a elemento,
// desde cualquier programa que la use, p.e. para usar los elementos uno a uno
// en algún cómputo, o visualizarlos en pantalla con algún formato específico,
// etc. El iterador no modifica la lista que recorre, sólo lee sus elementos.
// Para obtener un elemento cuenta con el método next(), que devuelve el elemento
// siguiente no visitado si existe, y si no, da una excepción. Por ello, antes
// de llamar a next() hay que comprobar si hay más elementos por visitar
// mediante hasNext() que devuelve true en caso afirmativo y false en caso
// contrario.

package aplicacionlistaarray;
import java.util.*;
/**
 *
 * @author bmoreno
 */

class Iterator {    //visibilidad desde el paquete
    // contiene una estructura (p.e. una lista), un elemento actual y
    // una forma de pasar al siguiente.

    private final ListaArray l;
    private int actual;    // actual es un índice al elemento actual

/**
 *
 * ------------------------------------------------------------------------
 * Constructor de un objeto iterador:
 *  - su lista (variable instancia) se inicializa a la lista que se desea
 *    recorrer y que es pasada como argumento al constructor, y
 *  - se inicializa el índice actual al del 1º elemento de la estructura para
 *    que el recorrido comience por el primer elemento.
 * @param lis, es la lista a recorrer. No hace falta que sea una copia de la
 * lista a recorrer, puede ser la original, pues no se va a modificar.
 */
    public Iterator(ListaArray lis){

        l = lis;    // podría clonarse lis (instanciando l con new y copiando
                    // lis en l) para trabajar con la copia, y no con el original
                    // pero no es necesario pues el iterador no modificará la
                    //lista.
        actual = 0;
    }

    /**-----------------------------------------------------------
     * Comprueba si quedan elementos por visitar en el iterador
     * @return true si quedan más elementos por visitar, false si no.
     */
    public boolean hasNext(){
        return (actual < l.size());
    }

    /**-------------------------------------------------------------------
     * Devuelve el siguiente elemento a visitar en la lista del iterador
     * y actualiza actual para que pase a indexar al siguiente elemento y
     * convertirlo en actual. La 1ª vez que se llama a next devuelve el 1º elem,
     * la 2º vez el 2º, etc.
     * Requisito: que exista un elemento por visitar. Si no existe lanzará
     * una excepcion. Por ello, el usuario de este método, antes de invocar
     * a next() debe comprobar que hasNext() da true.
     * @return
     * @throws NoSuchElementException
     */
    public int next() throws NoSuchElementException {
        int elem;

        if (actual < l.size()){
           elem = l.get(actual);
           actual++;
           return elem;
        }
```

```
            else
                throw new NoSuchElementException("Error en siguiente: no hay");
        }

        /**-------------------------------------------------------------
         * El iterador es reseteado para que comience un nuevo recorrido
         * desde principio.
         */
        public void reset(){
            actual = 0;
        }
    }
```