Parcial 2 – Tarea 5 – Generics
Jorge Ángel Rojas Martínez

Instructions

Precondiciones para esta tarea:

1. Interfaz List e Iterator ya existen
2. Clases LinkedList y ArrayList implementan su interfaz List
3. Clases ArrayListIterator y LinkedListIterator implementan su interfaz Iterator

Sobre el ejercicio de las listas hacer que List, Iterator, LinkedList, Node, LinkedListIterator, ArrayList y ArrayListIterator utilicen generics para definir su tipo de dato de modo que se puedan crear listas de cualquier tipo de dato con el operador diamante.

Modifiquen sus ejemplos para declarar e instanciar sus listas utilizando el operador diamante.

Como evidencia entregar PDF con lo siguiente:

1. Pantallazo de cada clase donde se vea la declaración de la clase o interfaz
2. Diagrama de clases (deben investigar cómo se representa la implementación de una interfaz)

ArrayList.java

```java
package uaslp.objetos.list.arraylist;

import uaslp.objetos.list.Iterator;
import uaslp.objetos.list.List;

4 usages
public class ArrayList <T> implements List<T>{
    23 usages
    private T []data;
    20 usages
    private int size;

    1 usage
    public ArrayList(){
        data = (T[])(new Object[2]);
    }

    no usages
    public ArrayListIterator<T> Iterator(){
        return new ArrayListIterator<>(this);
    }


    1 usage
    public void addAtTail(T data) {
        if(size == this.data.length){
            increaseArraySize();
        }
        this.data[size] = data;
        size++;

    }

    2 usages
    public void addAtFront(T data){
        if(size == this.data.length){
            increaseArraySize();
        }
        for(int i= size; i>0;i-- ){
            this.data[i] = this.data[i-1];
        }
```

ArrayList.java | ArrayListIterator.java

```java
package uaslp.objetos.list.arraylist;

import uaslp.objetos.list.Iterator;


2 usages
public class ArrayListIterator <T> implements Iterator {
    3 usages
    private ArrayList<T> arrayList;
    2 usages
    private int currentIndex=0;

    1 usage
    ArrayListIterator(ArrayList<T> arrayList) { this.arrayList = arrayList; }

    1 usage
    public boolean hasNext() { return currentIndex < arrayList.getSize(); }

    1 usage
    public T Next() { return (T) arrayList.getAt(currentIndex++); }
}
```

LinkedList.java

```java
package uaslp.objetos.list.linkedlist;

import uaslp.objetos.list.List;

1 usage
public class LinkedList <T> implements List<T> {
    9 usages
    private Node<T> head;
    6 usages
    private Node<T> tail;
    10 usages
    private int size;


    1 usage
    public void addAtTail (T data){
        Node<T> node = new Node<>();

        node.setPrevious(tail);
        tail = node;

        if(head == null){
            head = node;
        } else{
            node.getPrevious().setNext(node);
        }
        size ++;
    }


    2 usages
    public void addAtFront(T data) {

    }
```

LinkedList.java | LinkedListIterator.java

```java
package uaslp.objetos.list.linkedlist;

import uaslp.objetos.list.Iterator;

4 usages
public class LinkedListIterator <T> implements Iterator {
    6 usages
    private Node<T> iterator;

    1 usage
    LinkedListIterator(Node<T> node) { this.iterator = node; }

    1 usage
    public boolean hasNext() { return iterator.getNext() != null; }

    1 usage
    public T Next(){
        if(iterator.getNext() != null) {
            T data = iterator.getData();
            iterator = iterator.getNext();
            return data;
        }else {
            return null;
        }
    }
}
```

Node.java

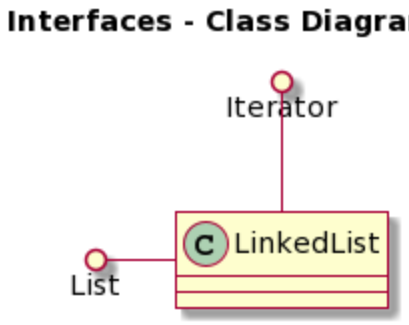
```java
package uaslp.objetos.list.linkedlist;

18 usages
public class Node <T>{
    2 usages
    T data;
    2 usages
    Node<T> next;
    2 usages
    Node<T> previous;

    3 usages
    T getData() { return data; }

    1 usage
    void setData(T data) { this.data = data; }

    7 usages
    public Node getNext() { return next; }

    3 usages
    public void setNext(Node next) { this.next = next; }

    4 usages
    public Node getPrevious() { return previous; }

    3 usages
    public void setPrevious(Node previous) {
        this.previous = previous;
    }
}
```

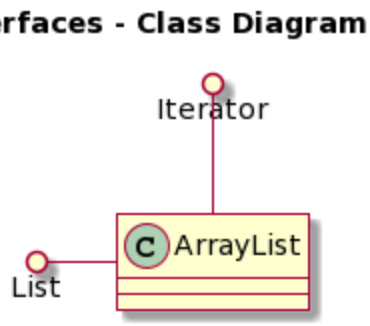
Iterator.java

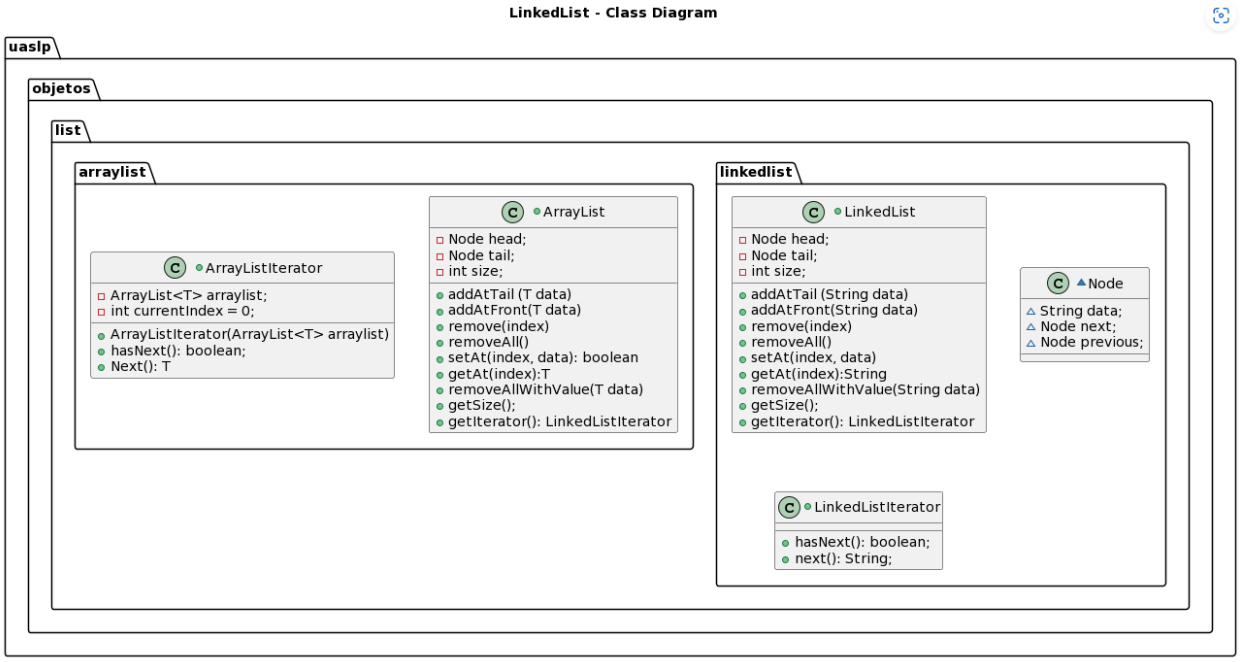
```java
package uaslp.objetos.list;

8 usages  2 implementations
public interface Iterator <T> {
    1 usage  2 implementations
    boolean hasNext();
    1 usage  2 implementations
    T Next();
}
```

List.java

```java
package uaslp.objetos.list;

6 usages  2 implementations
public interface List <T>{
    1 usage  2 implementations
    void addAtTail (T data);
    2 usages  2 implementations
    void addAtFront(T data);
    no usages  2 implementations
    boolean remove(T index);
    no usages  2 implementations
    void removeAll();
    no usages  2 implementations
    abstract boolean setAt(int index, T data);
    2 usages  2 implementations
    T getAt(int index);
    no usages  2 implementations
    void removeAllWithValue(T data);
    2 usages  2 implementations
    int getSize();
    1 usage  2 implementations
    Iterator<T> getIterator();
}
```


Interfaces - Class Diagram
Iterator
LinkedList
List


Interfaces - Class Diagram
Iterator
ArrayList
List


LinkedList - Class Diagram
uaslp
objetos
list
arraylist
ArrayListIterator
ArrayList<T> arraylist;
int currentIndex = 0;
ArrayListIterator(ArrayList<T> arraylist)
hasNext(): boolean;
Next(): T
ArrayList
Node head;
Node tail;
int size;
addAtTail (T data)
addAtFront(T data)
remove(index)
removeAll()
setAt(index, data): boolean
getAt(index):T
removeAllWithValue(T data)
getSize();
getIterator(): LinkedListIterator
linkedlist
LinkedList
Node head;
Node tail;
int size;
addAtTail (String data)
addAtFront(String data)
remove(index)
removeAll()
setAt(index, data)
getAt(index):String
removeAllWithValue(String data)
getSize();
getIterator(): LinkedListIterator
Node
String data;
Node next;
Node previous;
LinkedListIterator
hasNext(): boolean;
next(): String;