Corso di Programmazione Orientata Agli Oggetti

Progetto: Lista

È definita la seguente interfaccia relativa ad una lista di oggetti, liberamente ispirata a LinkedList<T> di java.util in cui i metodi omonimi conservano la stessa semantica:

```
public interface Lista<T> extends Iterable<T>{
        int size();
        void clear();
        void addFirst( T elem );
        void addLast( T elem );
        T getFirst();
        T getLast();
        T removeFirst();
        T removeLast();
        boolean isEmpty();
        boolean isFull();
        void sort( Comparator<T> c );
        ListIterator<T> listIterator();
        ListIterator<T> listIterator( int start );
}//Lista
```

Progettare e sviluppare:

- una classe astratta ListaAstratta<T>, che implementa l'interfaccia Lista<T> e concretizza quanti più metodi è possibile, e sicuramente equals(), toString(), hashCode()
- una classe concreta ListaConcatenata<T> erede di ListaAstratta<T> e basata su puntatori espliciti. In particolare, la classe deve realizzare una lista doppiamente concatenata (ogni nodo ha il puntatore al successore (next) e al predecessore (prior)) in modo da essere "navigabile" indifferentemente in avanti o indietro.

Rilevante nella classe ListaConcatenata<T> è l'implementazione del ListIterator<T>. A questo proposito si suggerisce di introdurre una inner class di ListaConcatenata<T>, diciamola IteratoreLista, che implementa ListIterator<T> e che può essere indifferentemente istanziata per fornire un Iterator<T> o un ListIterator<T> sulla lista.
Ai fini della realizzazione della struttura di iterazione, occorre riprodurre il fatto che l'iteratore si possa trovare prima del primo elemento della lista, dopo l'ultimo elemento, o fra due elementi consecutivi così come suggerisce la semantica base dell'iteratore. A questo scopo si può implementare il cursore dell'iteratore come un oggetto che mantiene al suo interno i riferimenti ai due nodi della lista "in mezzo" ai quali si trova l'iteratore. Tutto ciò, unitamente alla memorizzazione dell'ultimo movimento (next/previous) dell'iteratore (o assenza di movimento), potrebbe consentire agevolmente di portare a termine le operazioni remove/add nonchè l'inversione della direzione di iterazione da in avanti a indietro e viceversa.
Il metodo sort() riceve un oggetto Comparator<T> e ordina, ad esempio con l'algoritmo bubble sort, la lista. Si nota che quando uno scambio è richiesto tra i contenuti di due nodi, è conveniente scambiare solo le informazioni dei nodi, ossia è preferibile non modificare i puntatori durante gli scambi.
Successivamente si deve realizzare una GUI che espone una struttura a menù capace di evocare tutte le operazioni di Lista, e che dunque possa consentire all'utente di lavorare su una lista (es. di interi o di stringhe) e mostrare su una text area presente al centro della GUI lo stato corrente della lista, es. come [12, 4, 5, 15]. Quando si chiede di inserire un elemento (o rimuoverlo etc.) subito dopo occorre visualizzare sulla text area il nuovo contenuto della lista. Quando si accende un iteratore, si mostra la posizione (^) dove si trova l'iteratore, facendo next si mostra la nuova posizione mentre in un campo di testo Corrente si mostra l'elemento corrente

o ? se esso non è definito. Si nota che scegliendo il comando iterator, la freccia si trova inizialmente prima del primo elemento, e risultano abilitati i soli comandi hasNext, next, remove. Se invece si accende un ListIterator, si può specificare la posizione iniziale del list iterator come quella di default (identica a quella di iterator) o si può fornire un indice intero (compreso tra 0 e size()) che individua l'elemento della lista prima del quale va piazzata la freccia (specificando size(), la freccia va posta dopo l'ultimo elemento, specificando size()-1, la freccia va posta tra penultimo ed ultimo etc.).
Si suggerisce di introdurre una barra di menù con il menu File (con gli item New, Apri, Salva, Salva con nome, Exit), il menu Command (con gli item corrispondenti a tutte le possibili operazioni sulla lista), qualche item potrebbe aprire un menu di secondo livello (es. iterator, listIterator etc), etc.
Una parte rilevante del progetto è il suo testing estensivo. A questo riguardo si suggerisce di sviluppare e validare il codice Java senza la GUI. Quando si ritiene di aver ottenuto una soluzione corretta, allora si può passare a progettare e sviluppare l'interfaccia utente grafica. Si nota, infine, che limitatamente ai metodi disponibili, ListaConcatenata<T> deve comportarsi identicamente a java.util.LinkedList<T>.