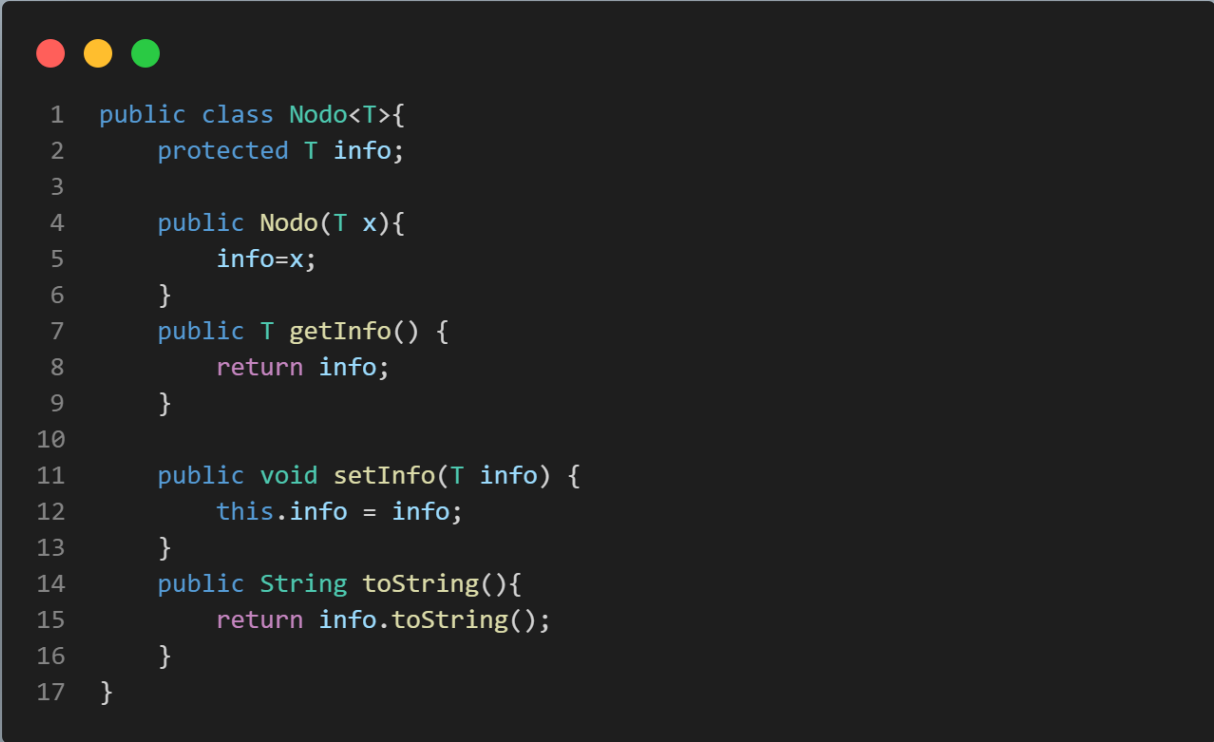


AlberoLF

AlberoLF è un albero con le seguenti caratteristiche:


1. ogni nodo contiene una sola informazione di cui non sappiamo il tipo e tutte le informazioni contenute nei nodi sono dello stesso tipo;
2. non è noto il numero massimo di figli che ogni nodo può avere;
3. l'inserimento di un nodo nell'albero (esclusa la radice) è possibile solo specificando il padre del nodo che verrà inserito e l'informazione che esso conterrà;
4. quando si inserisce un nodo "v" come figlio di un altro nodo "u" già presente nell'albero, il nodo "v" diviene l'ultimo figlio di "u".

NodoLF



```
1  public class Nodo<T>{
2      protected T info;
3
4      public Nodo(T x){
5          info=x;
6      }
7      public T getInfo() {
8          return info;
9      }
10
11     public void setInfo(T info) {
12         this.info = info;
13     }
14     public String toString(){
15         return info.toString();
16     }
17 }
```

La classe è astrazione del nodo di un albero, dove ogni nodo ha una informazione e i suoi metodo getter and setter



```
1  public class NodoLF<T> extends Nodo<T>{
2      private NodoLF<T> padre;
3      private LinkedList<NodoLF<T>> figli;
4
5      public NodoLF(T x){
6          super(x);
7          figli = new LinkedList<NodoLF<T>>();
8      }
9
10     public void addFiglio(NodoLF<T> v) {
11         figli.add(v);
12     }
13
14     public void setPadre(NodoLF<T> padre) {
15         this.padre = padre;
16     }
17
18     public NodoLF<T> getPadre(){
19         return padre;
20     }
21
22     ...metodi
23 }
```

La classe **NodoLF** è figlia della classe **Nodo** e aggiunge i suoi metodi

```
1 public class AlberoLF<T> {  
2     private NodoLF<T> radice;  
3  
4     ...metodi  
5 }
```

La classe **AlberoLF** è la amministratrice di tutti nodiLF.

```
1 public LinkedList<T> getInformazioniBFS() {  
2     LinkedList<NodoLF<T>> nodi = getNodi();  
3     LinkedList<T> informazioni = new LinkedList<T>();  
4     for (NodoLF<T> nodoLF : nodi) {  
5         informazioni.add(nodoLF.getInfo());  
6     }  
7     return informazioni;  
8 }
```

Il metodo **getInformazioniBFS** restituisce la lista delle informazioni dei nodi dell'albero visitato in ampiezza. Prende i nodi con il metodo BFS e memorizza la lista delle informazioni.

Il metodo **getInformazioniDFS** si comporta allo stesso modo del metodo precedente, in più visita i nodi con la visita profondità.

Visita in ampiezza

```
1 public LinkedList<NodoLF<T>> bfs(NodoLF<T> radice) {
2     LinkedList<NodoLF<T>> queue = new LinkedList<NodoLF<T>>();
3     LinkedList<NodoLF<T>> listaNodi = new LinkedList<NodoLF<T>>();
4
5     queue.add(radice);
6
7     while (!queue.isEmpty()) {
8         NodoLF<T> u = queue.remove();
9         queue.addAll(u.getfigli());
10        listaNodi.add(u);
11    }
12    return listaNodi;
13 }
```

Visita in profondità


```
1 public LinkedList<NodoLF<T>> dfs() {
2     LinkedList<NodoLF<T>> listaNodi = new LinkedList<NodoLF<T>>();
3     dfs1(radice, listaNodi);
4     return listaNodi;
5 }
6
7 public void dfs1(NodoLF<T> nodo, LinkedList<NodoLF<T>> listaNodi) {
8     listaNodi.add(nodo);
9     for (NodoLF<T> nodoLF : nodo.getfigli()) {
10        dfs1(nodoLF, listaNodi);
11    }
12 }
```

Stampa

```
1  public class StampaAlbero<T> {
2      private AlberoLF<T> albero;
3
4      public StampaAlbero(AlberoLF<T> albero) {
5          this.albero = albero;
6      }
7
8      public void printAll() {
9          printRadice();
10         printRappresentazione();
11         printInformazioni();
12         printAltezza();
13     }
14
15
16     public void printRadice() {
17         System.out.println("radice dell'albero: "+albero.getRadice());
18     }
19
20     ...metodi
21 }
```

StampaAlberoLF stampa tutta la informazioni dell'albero.

Test



```
1  class AlberoLF_Test {
2
3      AlberoLF<Integer> albero = new AlberoLF<Integer>();
4
5      @Test
6      public void radiceTest() {
7          albero.addRadice(7);
8          NodoLF<Integer> radice = albero.getRadice();
9          int info = radice.getInfo();
10         assertEquals(7, info);
11     }
12
13     @Test
14     public void radiceNuovaTest() {
15         albero.addRadice(7);
16         NodoLF<Integer> radiceVecchia = albero.getRadice();
17         albero.addRadice(5);
18         NodoLF<Integer> radice = albero.getRadice();
19         NodoLF<Integer> padre = radiceVecchia.getPadre();
20         assertEquals(padre, radice);
21     }
22
23     ...metodi
24 }
```

La classe AlberoLF_Test è composta dai test per controllare il giusto funzionamento della classe AlberoLF.