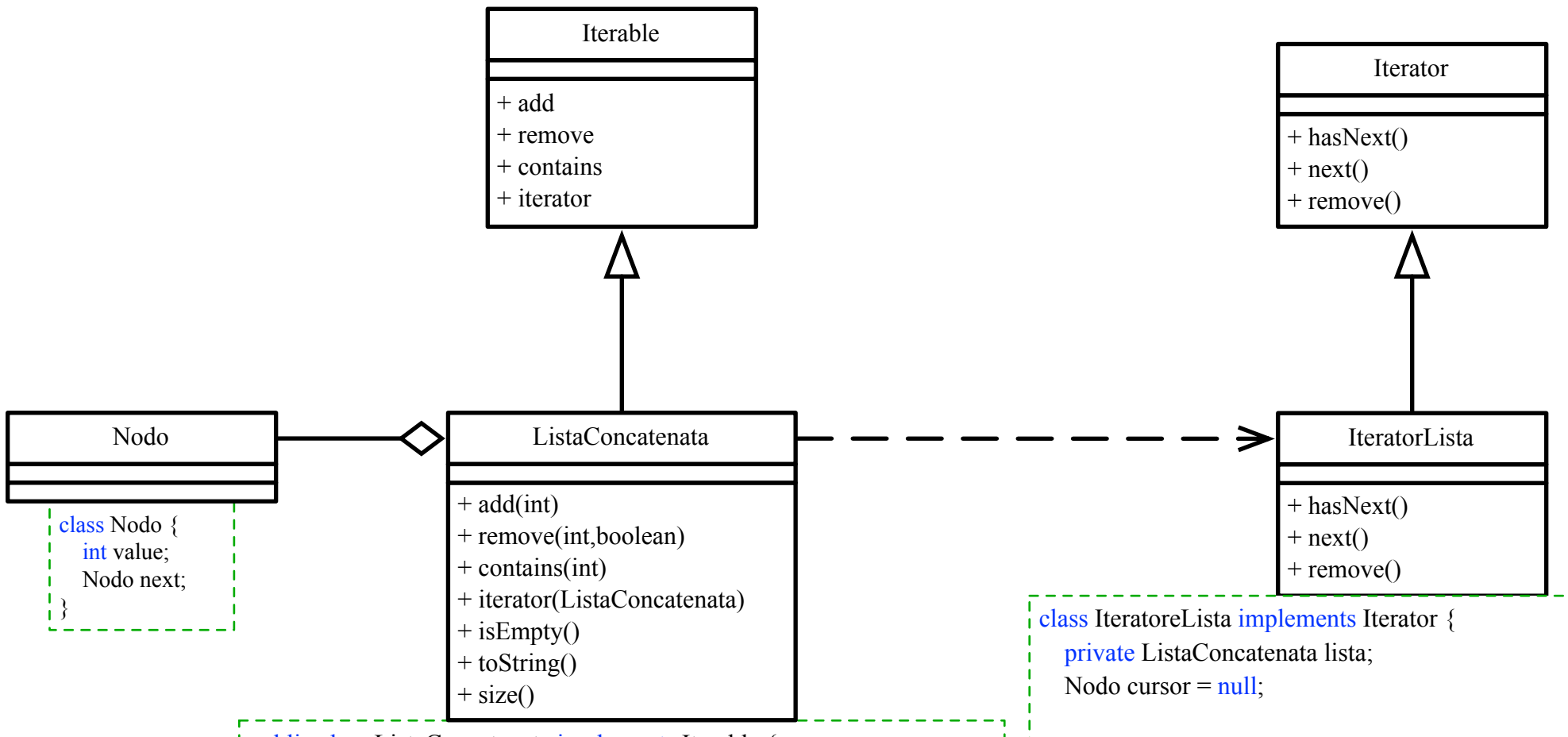


```
class Nodo {
    int value;
    Nodo next;
}
```

```
class IteratoreLista implements Iterator {
    private ListaConcatenata lista;
    Nodo cursor = null;

    public IteratoreLista(ListaConcatenata lista) {  // Crea un Iteratore che gestisce la “ListaConcatenata”
        this.lista = lista;
        cursor = lista.start;
    }

    public boolean hasNext() {
        return cursor!=null;
    }

    public Object next() {
        int value = cursor.value;
        cursor = cursor.next;
        return value;
    }

    public void remove() {
        throw new UnsupportedOperationException("Not supported yet.");
    }
}
```

```
public class ListaConcatenata implements Iterable {
    Nodo start;
    int size;

    public ListaConcatenata() {  // Costruttore
        size = 0;
        start = null;    }

    public void add(int n) {
        Nodo nuovo = new Nodo();
        nuovo.value = n;
        if (start==null) {  // Lista vuota
            start = nuovo;
            start.next = null;
        } else if (start.next==null) {  // Lista con un solo nodo
            start.next = nuovo;
        } else {  // Lista con più nodi
            Nodo iter = null;
            for(iter=start.next; iter.next != null; iter = iter.next);
            iter.next = nuovo;
        }
        size++;    }

    public void remove(int value, boolean all) {  // “all” = true; cancella tutti gli elementi della lista uguali a “value”, altrimenti concella solo il primo che trova.
        while(start!=null && start.value==value) {  // Se all’inizio della lista ho più di un nodo uguale a “value”, con il while cancello tutti i nodi iniziali uguali a value.
            start = start.next;
            size--;
            if (!all || start==null) return;
        }
        Nodo current = start.next;
        Nodo previous = start;
        for(; current!=null; current=current.next) {
            if(current.value == value) {
                previous.next = current.next;
                current = previous;
                size--;
                if(!all) return;  // Se dovevo cancellarne solo uno, esco.
            } else {
                previous = previous.next;
            }
        }    }

    public boolean isEmpty() {
        return size==0;    }

    public String toString() {  // Restituisce il nodo come stringa “{nodo}”
        String str = "{";
        for(Object element:this)
            str+=element+",";
        str += "}";
        return str;    }

    public int size() {
        return size;    }

    public boolean contains(int n) {
        for(Nodo iter = start; iter!=null; iter = iter.next)
            if (iter.value==n) return true;
        return false;    }

    public Iterator iterator() {
        return new IteratoreLista(this);    }
}
```