Sebbene l'interfaccia specifichi il metodo remove(); lasciamo l'implementazione del metodo è lasciata in bianco. Ci comportiamo in questo modo perché è sempre meglio non modificare la collezione durante l'iterazione.

Linked Lists

Definizione: Una linked list è una struttura dati ricorsiva che può essere vuota (Null) oppure una reference (riferimento) ad un nodo avente un item generico ed una reference ad una linked list.

Un NODO in questa definizione è un'entità astratta che può contenere qualsiasi tipo di valore (DATA), in aggiunta al riferimento al nodo che caratterizza il suo ruolo nella creazione delle linked lists.

Node record: Con i linguaggi ad oggetto, implementare delle linked lists non è difficile. Iniziamo con una classe annidata che definisca l'astrazione del nodo.

```
private class Node {
        Item item;
        Node next;
    }
```

Un nodo ha due variabili: un Item ed un Node.

Iterazione

Uno delle operazioni sulle collezioni è quella di processare ogni elemento iterando attraverso la collezione usando lo statement di java Foreach:

```
For (String s: collection)
        StdOut.println(s);
```

Questo codice è equivalente a:

```
Iterator<String> i = collection.iterator();
While (i.hasNext()) {
:   String s = i.next();
}
```

Il codice esprime gli ingredienti necessari per l'uso dell'iterator:

- la collezione deve implementare iterator()
- la classe Iterator deve includere i metodi:
  - hasNext() che ritorna un boolean, e il metodo
  - next() che ritorna un elemento generico della collez.

Che cosa è un Iterator?
È un oggetto che implementa i metodi hasNext() e Next(), definito dall'interfaccia:

```
public interface Iterator<Item> {
:   boolean hasNext();
:   Item next();
:   void remove();
}
```

Definiamo Node nella stessa classe in cui vogliamo usarlo, e lo creiamo come private perché non vogliamo che sia utilizzabile dal client.

Come con qualsiasi Data Type, creiamo l'oggetto di Tipo Node invocando il costruttore, senza argomenti, new Node().
Il risultato è una reference ad un oggetto Node, le quali var. d'istanza sono entrambe inizializzate a Null.

'Costruire' una linked list
Possiamo rappresentare una LinkedL. con variabili di Tipo Node, semplicemente assicurandoci che il suo valore sia o Null o un riferimento all'elemento successivo.
Creiamo un nodo per ogni elemento:

```
Node first = new Node();
Node second = new Node();
Node third = new Node();
```

Ed impostiamo il valore di Item per ognuno:

```
first.item = 'to';
second.item = 'be';
third.item = 'or';
```

Ed in fine il riferimento all'elemento succ:

```
first.next = second;
second.next = third;
```

Inserimento in testa

Supponiamo di voler inserire un elemento in una linked.list. Il posto più semplice dove inserirlo è all'inizio della lista.
Inoltre, è un'operazione che richiede sempre lo stesso numero di operazioni, che non dipende dalla lunghezza della lista.

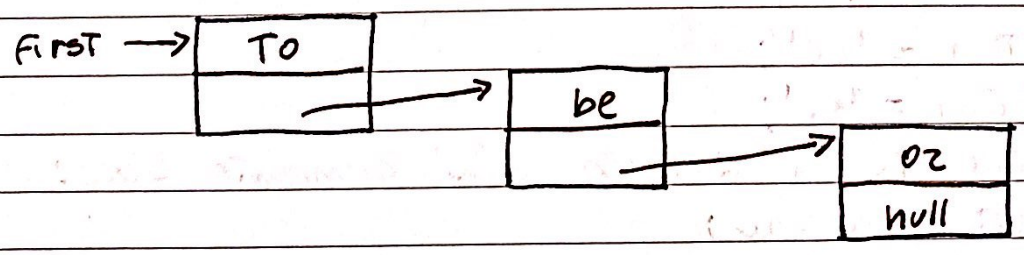
Ci basterà creare il nuovo nodo, e 'linkarlo' al 'primo' nodo precedente.

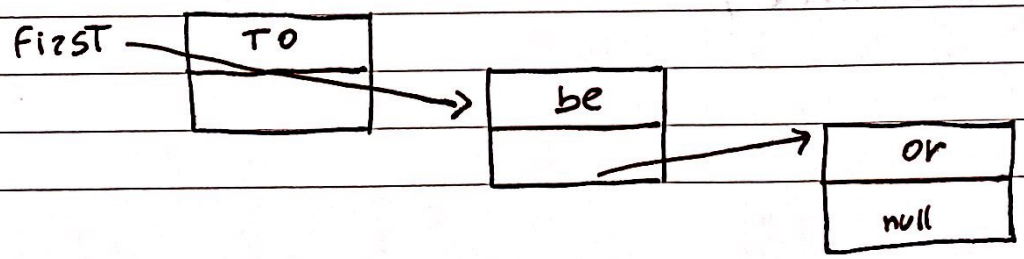
```
Node oldFirst = first;   salvo il vecchio primo nodo
first = new Node();      creo il nuovo nodo
first.item = "new item"; assegno il valore
first.next = oldfirst;   assegno il nodo succ (oldfirst)
```

Rimozione dalla testa

Pg 145

Per questa azione basta assegnare a first il valore di first.next. Tipicamente, il nodo diventerebbe orfano, ed il garbage collector di Java pulirebbe la memoria occupata.





Inserimento in coda

Per fare ciò, abbiamo bisogno di un link all'ultimo elemento nella lista, perché deve essere cambiato in modo da puntare al 'nuovo' ultimo nodo.

```
Node oldLast = last;        salvo il vecchio nodo
Node last = new Node();     creo il nuovo ultimo nodo
last.item = "not";          assegno il valore
oldLast.next = last;        assegno il rif. all'ultimo
                            nuovo elemento.
```