



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA



DISIM
Dipartimento di Ingegneria
e Scienze dell'Informazione
e Matematica



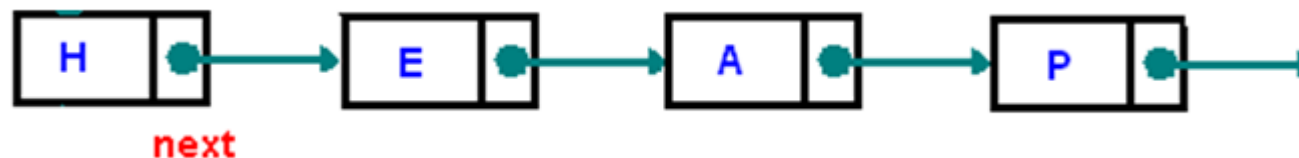
Laboratorio di Algoritmi e Strutture Dati a.a. 2024/2025

Introduzione alla classe
`SinglyLinkedList<E>`:
una classe giocattolo

Giovanna Melideo
Università degli Studi dell'Aquila
DISIM

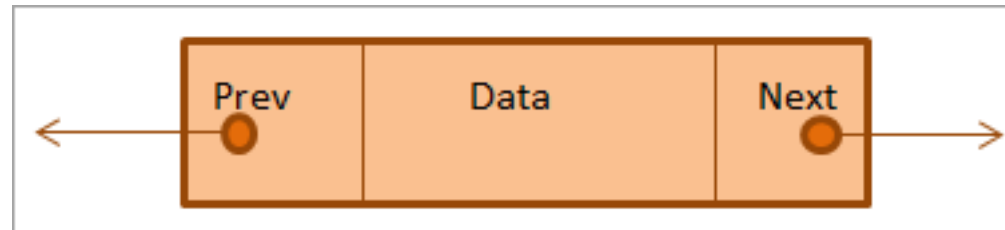
Lista concatenata

- Una lista concatenata “semplicemente collegata” è un oggetto di tipo `List` (cioè un esemplare di una classe che implementa l'interfaccia `List`) che soddisfa la seguente proprietà:
 - Ciascun elemento della lista (es: **H**, **E**, **A**, **P**, ...) è contenuto in un oggetto di tipo `Node`, (“nodo”) che contiene almeno un collegamento (**link**) all'oggetto di tipo `Node` contenente l'elemento successivo (next) della lista
 - L'oggetto di tipo `Node` che contiene l'ultimo elemento della lista non ha un elemento «successivo»



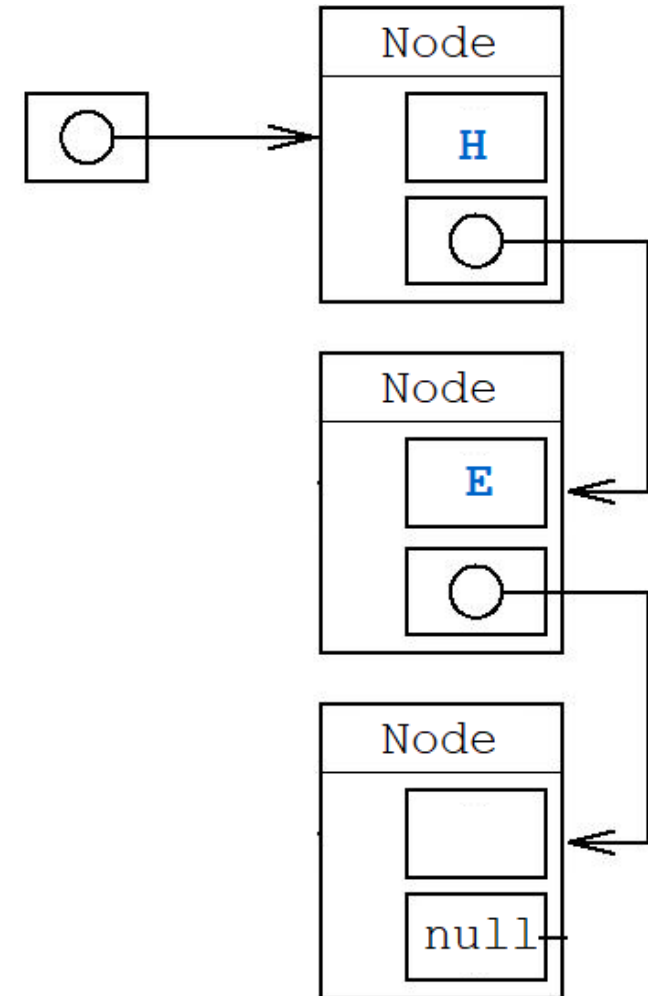
Lista doppiamente concatenata

- Alcune liste, dette “doppiamente collegate” soddisfano anche la seguente proprietà:
 - Ciascun oggetto di tipo **Node** contiene anche un collegamento all’oggetto di tipo **Node** contenente il precedente (previous) elemento della lista



SinglyLinkedList: una classe "giocattolo"

- Una lista semplicemente collegata è una struttura dati dinamica lineare in cui ogni nodo è un oggetto separato



Nested/inner classes (richiami)

- Una classe interna (nested class) è una classe la cui definizione è interna a quella di un'altra classe.
- Sono divise in due categorie:
 - **static nested classes**: le classi interne dichiarate con la parola chiave `static` non sono collegate ad un'istanza ma possono accedere solo alle variabili e ai metodi statici della classe esterna.
 - **non-static nested classes** (o inner classes): le classi interne non statiche sono associate alle singole istanze e possono accedere alle variabili e ai metodi della classe esterna, e viceversa.
- Nota: la decisione di statico o non statico dipende principalmente dal fatto che sia necessario o meno poter accedere direttamente ai campi e ai metodi della classe esterna

Inner classes (richiami)

Un esempio di accesso ad un membro della classe esterna:

```
class ClasseEsterna {  
    private int x = 9;  
    class ClasseInterna {  
        public void stampaX() {  
            System.out.println("Il valore di x è " + x);  
        }  
    } //end-ClasseInterna  
} //end-ClasseEsterna
```

- Il codice precedente stamperà il numero 9 poiché l'inner Class è in grado di accedere ai membri dell'Outer Class in maniera diretta anche se dichiarati privati.

Linked Lists: la classe Node

In seguito definiamo una classe `SinglyLinkedList` con due classi interne (rif. **`SinglyLinkedList.java`**):

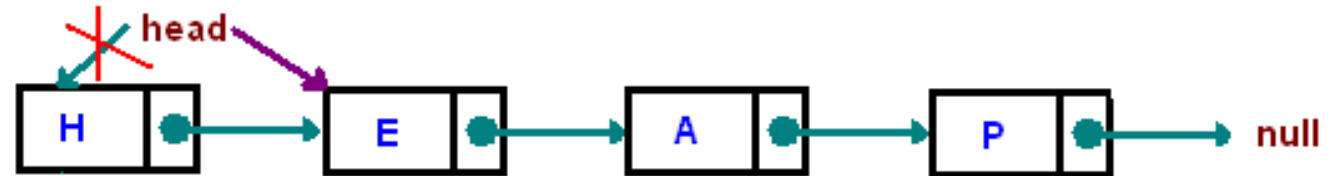
- static **`Node`** class and
- non-static **`SinglyLinkedListIterator`** class

```
private static class Node<E> {  
    private E data;  
    private Node<E> next;  
  
    public Node(E data, Node<E> next) {  
        this.data = data;  
        this.next = next;  
    }  
}
```

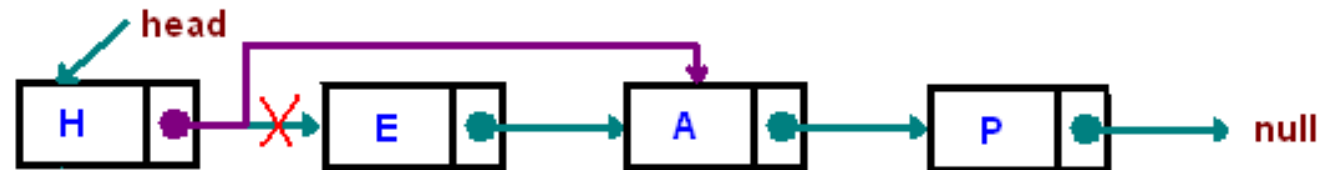
Esempi

```
Node head = new Node<E> (...);
```

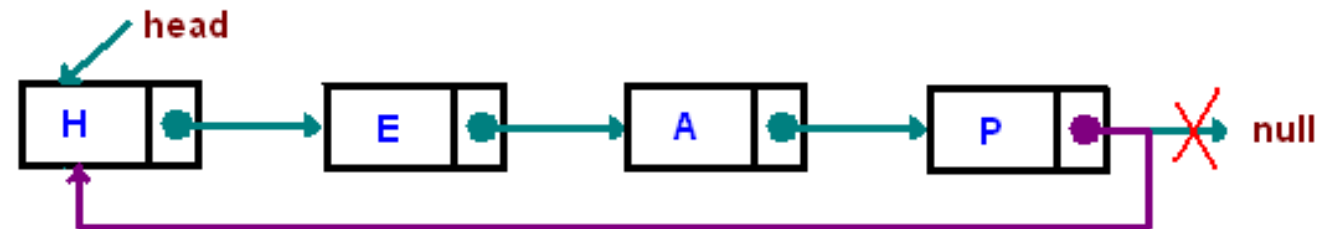
```
head = head.next;
```



```
head.next = head.next.next;
```



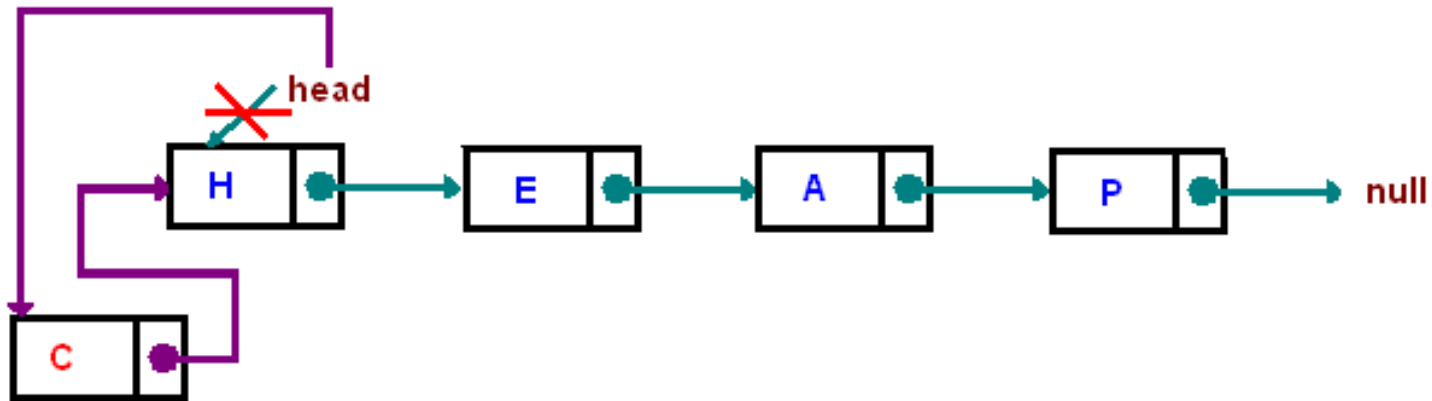
```
head.next.next.next.next =  
head;
```



Linked List (selected) operations

- **addFirst** crea un nodo e lo aggiunge all'inizio della lista:

```
public void addFirst(E item) {  
    head = new Node<E>(item, head);  
}
```

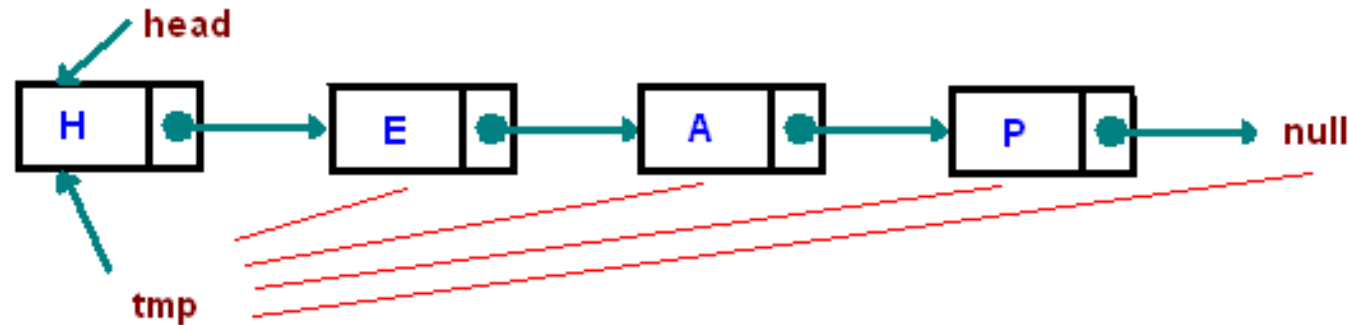


Linked List Operations: traversing

- Iniziando da **head** (senza cambiare il riferimento) si accede ad ogni nodo fino a quando non si raggiunge **null**.

```
Node tmp = head;
```

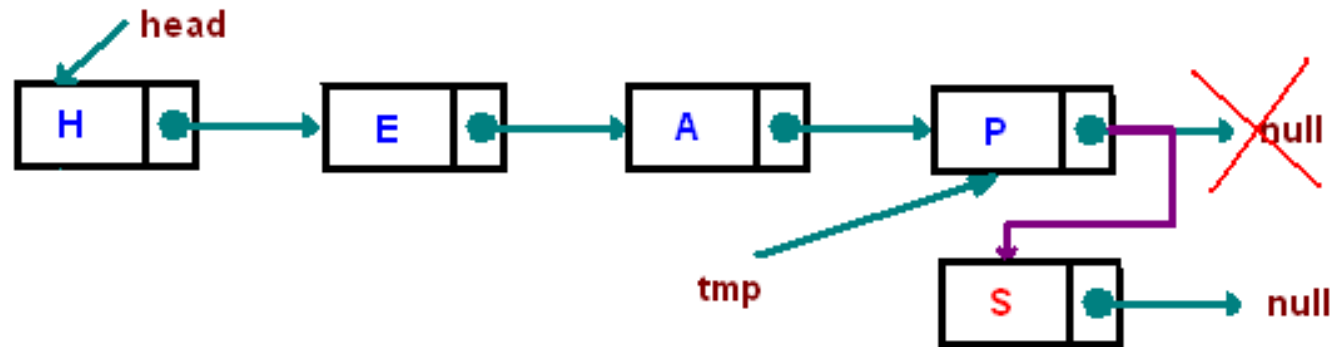
```
while (tmp != null) { ... tmp = tmp.next; }
```



Linked List Operations: addLast

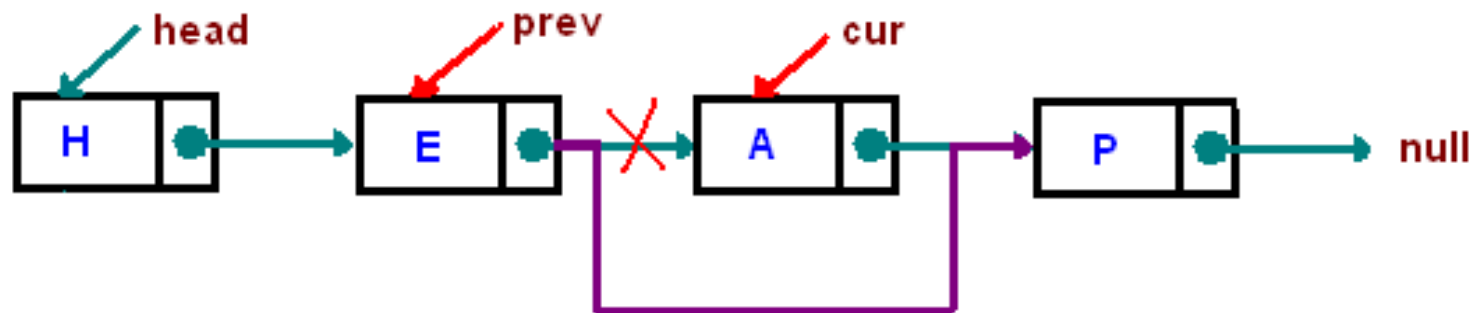
- **addLast** appende il nodo alla fine della lista al termine di un'iterazione. **Si può fare meglio?**

```
public void addLast(E item) {  
    if(head == null) addFirst(item);  
    else {  
        Node<E> tmp = head;  
        while(tmp.next != null) tmp = tmp.next;  
        Node<E> newNode = new Node<E>(item);  
        tmp.next = newNode;  
        newNode.next = null;  
    }  
}
```



Linked List Operations: **remove**

- **remove** cerca un nodo contenente l'oggetto "key" da cancellare e, se presente, lo cancella. Casi da gestire:
 - la lista è vuota
 - bisogna cancellare il primo nodo
 - bisogna cancellare un nodo in posizione intermedia
 - l'oggetto cercato non è in lista



Linked List Operations: iterator

```
public Iterator<E> iterator() {  
    return new SinglyLinkedListIterator();  
}
```

- **SinglyLinkedListIterator** è una classe privata interna alla classe `SinglyLinkedList`



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA



DISIM
Dipartimento di Ingegneria
e Scienze dell'Informazione
e Matematica



Domande?

Giovanna Melideo
Università degli Studi dell'Aquila
DISIM