PROGRAMMAZIONE 2

4. Un modello operazionale per Java

Abstract Stack Machine

- Abstract Stack Machine: modello computazionale per Java che permette di descrivere la nozione di stato modificabile
- Modello astratto: nella seconda parte del corso esamineremo nel dettaglio gli aspetti relativi alla realizzazione dei linguaggi di programmazione

Struttura

- Java ASM: tre componenti fondamentali
  - Workspace per la memorizzazione dei programmi in esecuzione
  - Stack per la gestione dei binding
  - Heap per la gestione della memoria dinamica (oggetti)
- Oltre a questi componenti la ASM è caratterizzata da uno spazio di memoria dinamica che serve per memorizzare le tabelle dei metodi degli oggetti
  - per semplicità ora non considereremo questa parte

Perche' queste strutture dati?

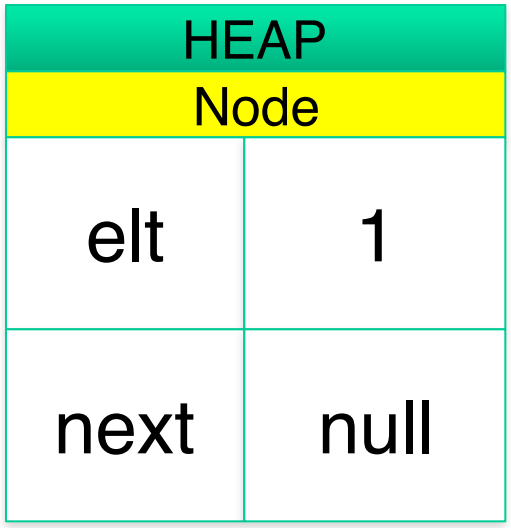
Abstract Stack Machine

- Buon modello per comprendere come funzionano i programmi Java
- Definisce in modo chiaro come sono gestite le strutture dati
- Permette di trattare in modo omogeneo la gestione degli oggetti
- Visione semplificata della macchina astratta per la realizzazione del linguaggio

Allocazione di oggetti sullo heap

```
class Node {
      private int elt;
      private Node next;
public Node (int e0, Node n0) {
       elt = e0;
       next = n0;  }
 }

Node n = new Node(1, null);
```

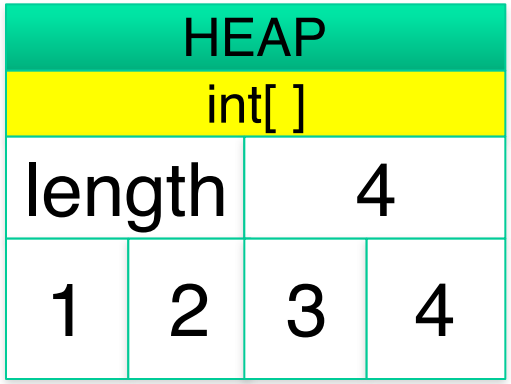


Le variabili di istanza possono essere mutabili o meno

Nota: a run-time sono presenti informazioni di tipo esemplificate dalla “annotazione di tipo” `Node` memorizzate nello heap. Il perché si capirà in seguito!!!

Allocazione di array sullo heap

```
int[] anArray;
anArray = new int[4];
anArray[0] = 1;
:
anArray[3] = 4;
```

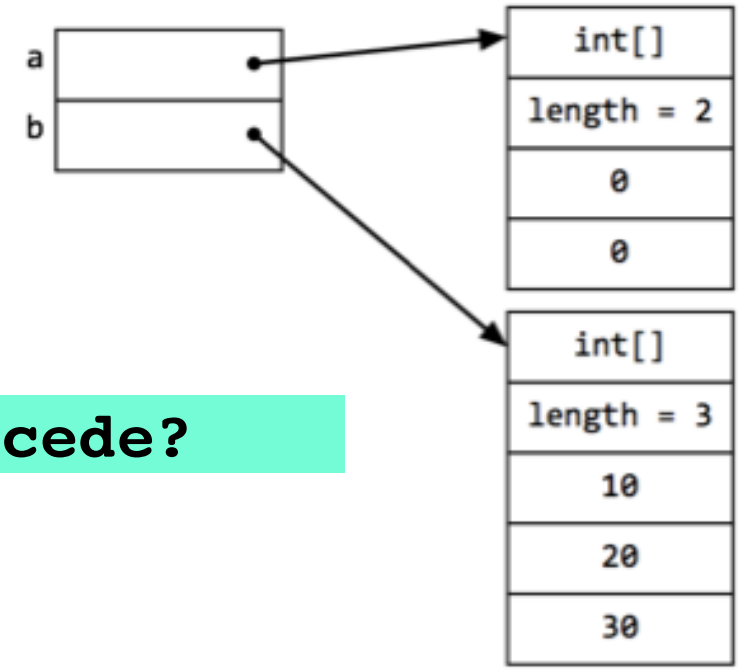


Il valore di **`length`** è fissato
Gli elementi dell'array sono mutabili

Nota importante: a run-time sono presenti informazioni di tipo esemplificate dal "tipo" array di interi e la dimensione (length) memorizzate nello heap

Esempio: array

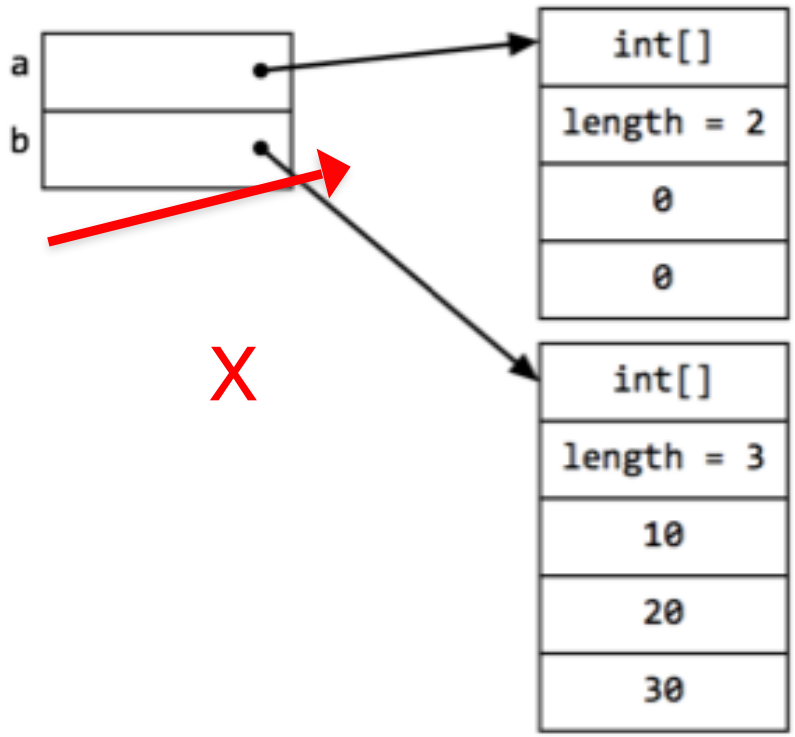
```
int[] a = new int[2];
int[] b = new int[] {10, 20,30};
```



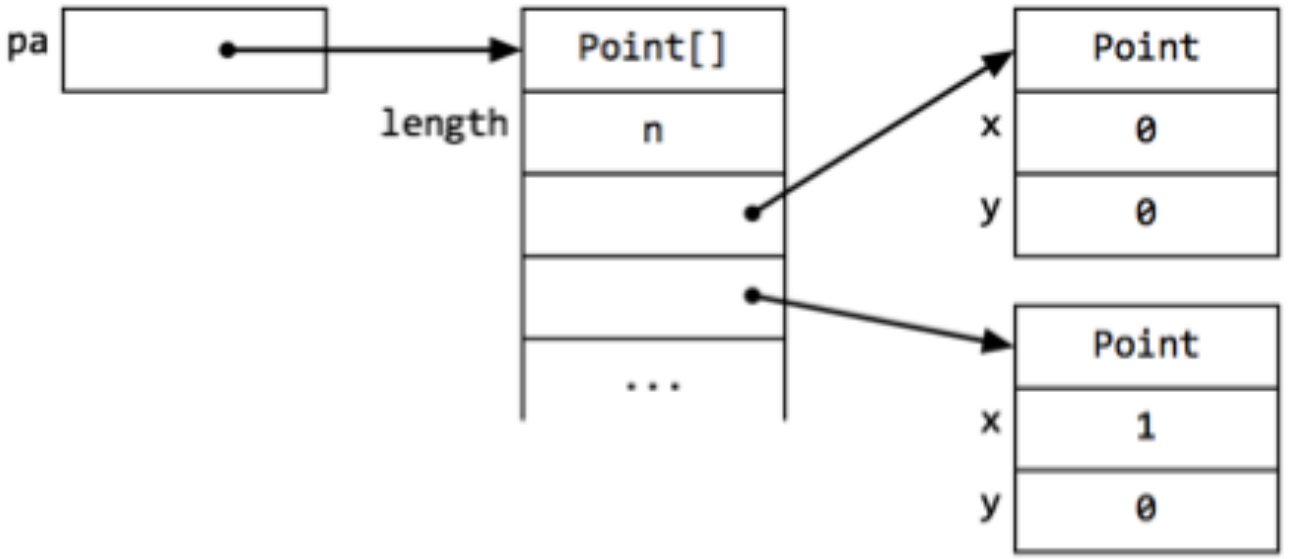

```
b=a   //Cosa succede?
```

Esempio: aliasing

a e b sono alias: riferimenti allo stesso oggetto!!!



```
class Point {
   int x, y;
 Point(int x, int y) {  this.x = x;  this.y = y; }
}
int n = ...;
Point[] pa = new Point[n];
for (int i = 0; i < n; i++) {
 pa[i] = new Point(i, 0);
```



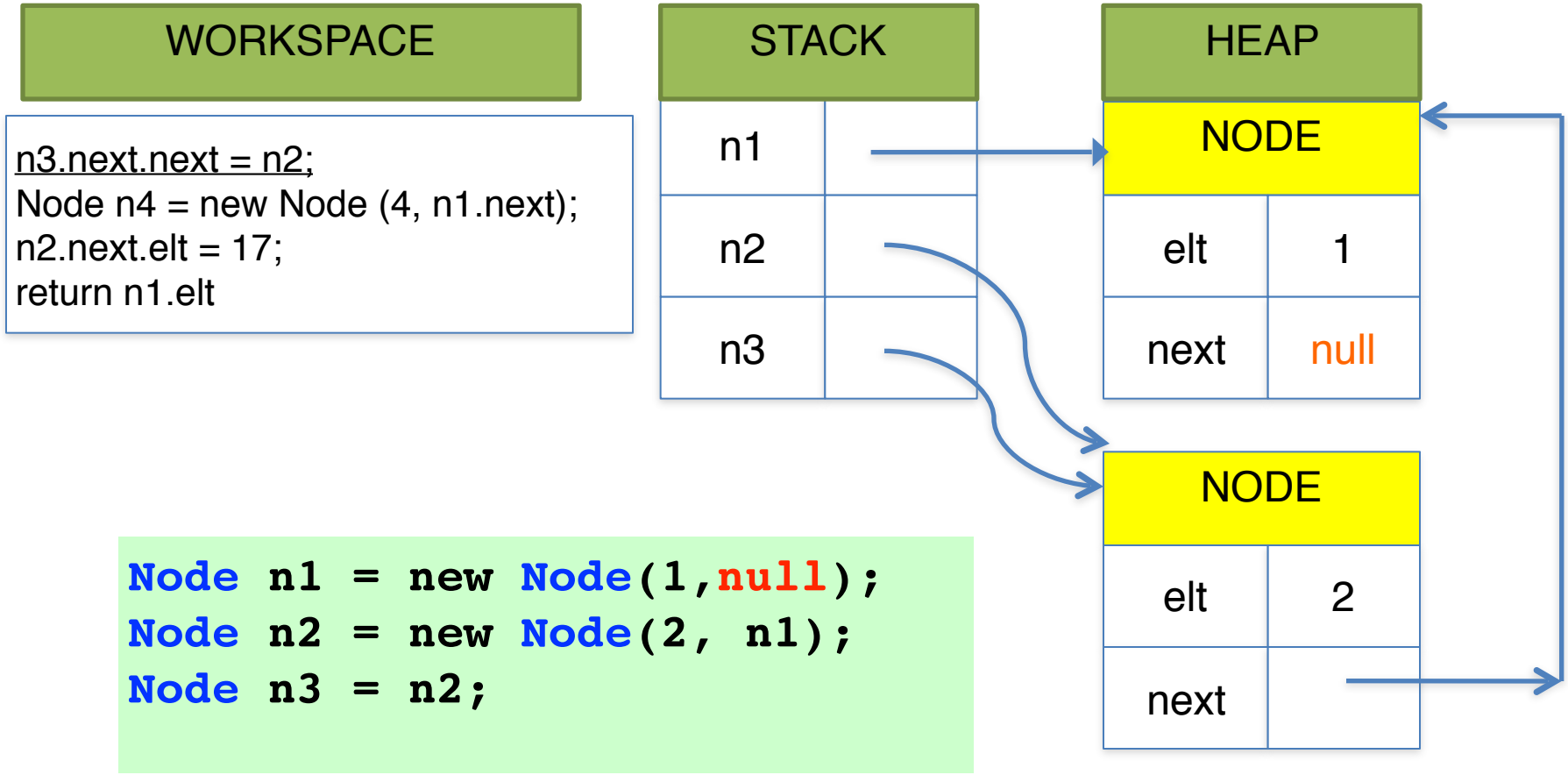
Esempio

```
class Node {
   public int elt;
   public Node next;
public Node(int e0,Node n0)
{
        elt = e0;
        next = n0;  }
 }
 :
}
```

```
public static int m( ) {
   Node n1 = new Node(1,null);
   Node n2 = new Node(2, n1);
   Node n3 = n2;
   n3.next.next = n2;
  Node n4 =new Node(4, n1.next);
   n2.next.elt = 17;
   return n1.elt;
}
```

Esempio di ASM

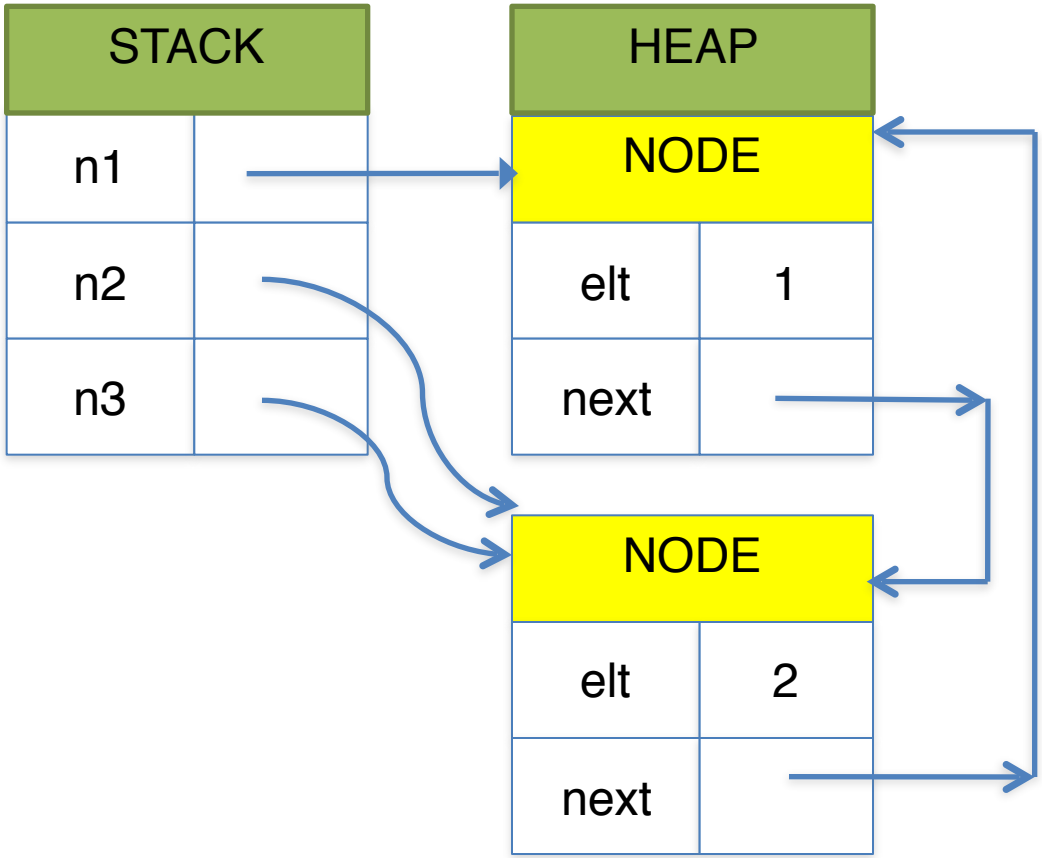
- Supponiamo di voler valutare l'invocazione del metodo statico
- La prima cosa da osservare è che l'invocazione del metodo statico restituisce un valore intero (che non è un oggetto)
- Lo stack deve contenere lo spazio per memorizzare il valore restituito dall'invocazione del metodo
  - intuitivamente una variable di tipo int
  - per semplicità lo omettiamo




```
Node n1 = new Node(1,null);
Node n2 = new Node(2, n1);
Node n3 = n2;
```

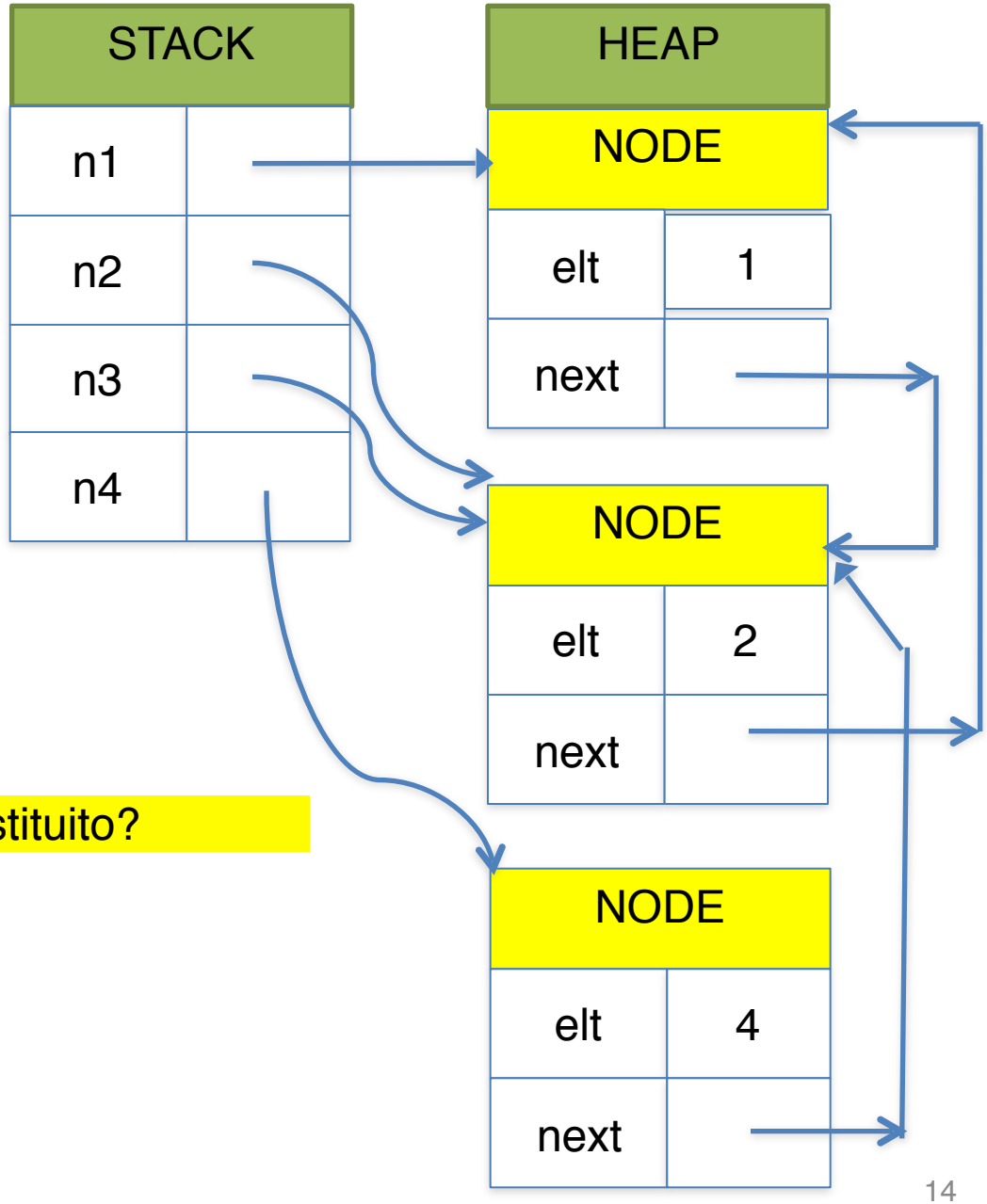
WORKSPACE

n3.next.next = n2;
Node n4 = new Node (4, n1.next);
n2.next.elt = 17;
return n1.elt



WORKSPACE

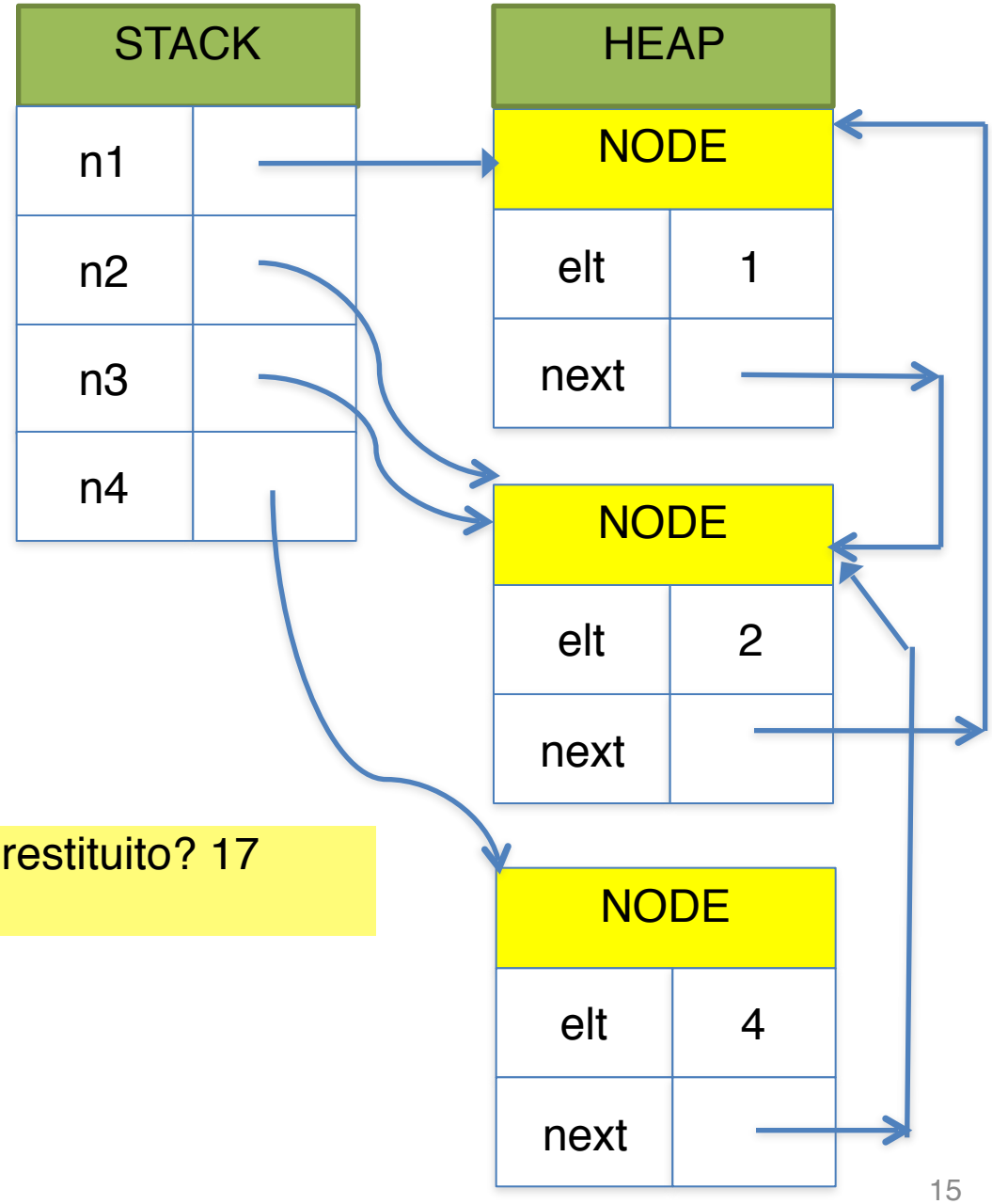
Node n4 = new Node (4, n1.next);
n2.next.elt = 17;
return n1.elt



Domanda: quale è il valore restituito?

WORKSPACE

Node n4 = new Node (4, n1.next);
n2.next.elt = 17;
return n1.elt



Domanda: quale è il valore restituito? 17
Come mai?