

# Programmazione ad Oggetti

## Progetto (Liste collegate)

V 1.2 © Marco Torchiano 2005

## Esercizio

- Costruire una lista di interi utilizzando il linguaggio C.
  - ♦ Insert: inserisce in testa
  - ♦ First: restituisce il primo elemento
  - ♦ Remove: rimuove il primo elemento
  - ♦ Size: restituisce la dimensione
  - ♦ Si ignori la gestione degli errori.

## Esempio di uso

```
#include "list.h"
int main(int argc, char* argv[]){
    init();
    insert(1);
    insert(2);
    insert(3);
    printf("Size: %d\n",size());
    printf("First: %d\n",first());
    delete();
    printf("Removed\n");
    printf("First: %d\n",first());
}
```

## Soluzione

È tutto giusto?

<pre>typedef struct el {     struct el* next;     int info; } element; element* head; void init(){     head = NULL; } void delete() {     head = head-&gt;next; } int first(){     return head-&gt;info; }</pre>	<pre>void insert(int i){     element* p=(element*)malloc...     p-&gt;next = head;     p-&gt;info = i;     head = p; } int size(){     int count=0;     element *p=head;     <del>while(p!=NULL) count++;</del>     return count; } for(;p!=NULL;p=p-&gt;next)     count++;</pre>
--	---

## Problemi

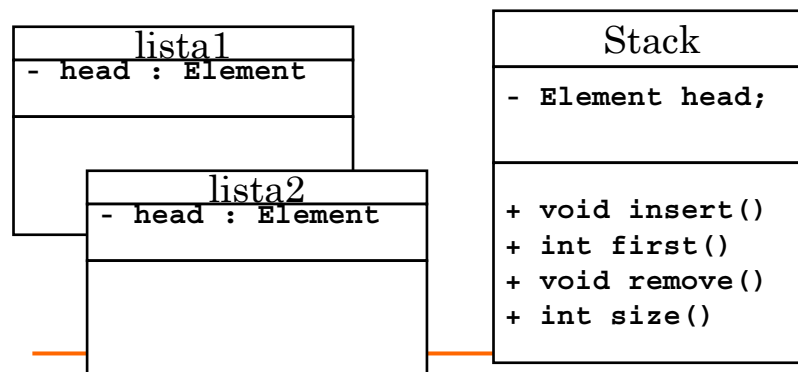
---

- Risolvere i seguenti problemi:
    - ♦ Come è possibile utilizzare due liste distinte all'interno dello stesso programma ?
    - ♦ Come è possibile garantire che la lista venga inizializzata prima di essere usata ?
    - ♦ Cosa succede se non inizializzo la lista?
    - ♦ Cosa restituisce first() quando la lista è vuota?
- 

## La classe List

---

- La classe List descrive una lista di interi.
- Ogni oggetto istanza della classe List realizza una lista distinta dagli altri.



## Esempio di Uso di List

---

```
public static void main(String[] Args){
    Console con = new Console();
    List list = new List();
    list.insert(1);
    list.insert(2);
    list.insert(3);
    con.println("Size: " + list.size());
    con.println("First: " + list.first());
    list.remove();
    con.println("Removed");
    con.println("First: " + list.first());
}
```

---

## Allocazione variabili, reference

---

### ▪Classi

- ♦Tutte le variabili oggetto sono dei puntatori
- ♦Le istanze sono allocate in memoria dinamica (operatore new)

```
List l; // NON alloca spazio, solo reference
l = new List() // alloca spazio, necessario
```

### ▪Tipi nativi (int, float, char, etc. )

```
int i; // alloca spazio
```

---

## == vs. equals()

---

- L'operatore == confronta i riferimenti
- Il metodo equals() confronta gli oggetti
- Es.

```
String s1 = "A";  
String s2 = new String("A");  
if(s1!=s2) // due oggetti distinti  
if(s1.equals(s2)) // stesso contenuto
```

---

## Progettazione della lista

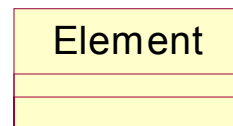
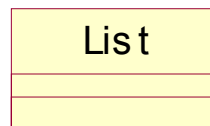
---

- Progettare utilizzando l'approccio ad oggetti una **linked list**
  - La lista contiene elementi di tipo intero
  - Deve essere possibile inserire dei nuovi elementi in testa alla lista
  - La lista deve fornire il numero di elementi presenti
-

## Identificazione classi

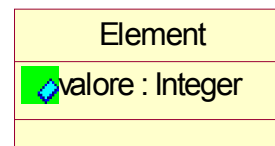
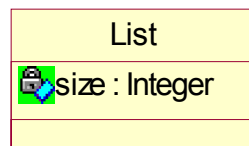
---

- In base alla nostra conoscenza sui tipi di dato astratti

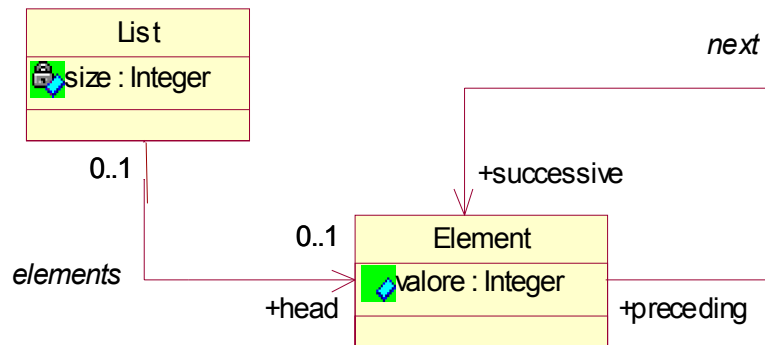


## Identificazione attributi

---



## Identificazione relazioni



## Identificazione metodi

