



UNIVERSITÀ  
DEGLI STUDI  
DELL'AQUILA



DISIM  
Dipartimento di Ingegneria  
e Scienze dell'Informazione  
e Matematica



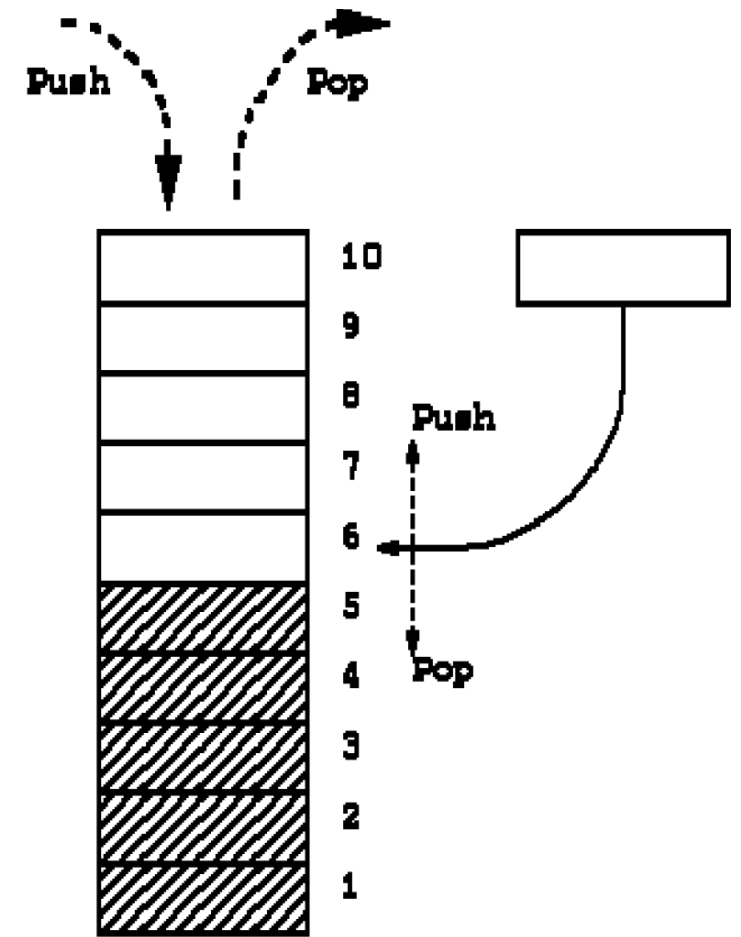
# Laboratorio di Algoritmi e Strutture Dati a.a. 2024/2025

Il tipo di dato Stack

**Giovanna Melideo**  
Università degli Studi dell'Aquila  
DISIM

# Tipo di dato Pila (Stack)

- Uno stack è una collezione di elementi dello stesso tipo che supporta le seguenti operazioni:
  - push, pop, peek o top, isEmpty, [isFull]
- Disciplina di accesso **LIFO** - last in first out: l'accesso agli elementi avviene secondo l'ordine inverso di inserimento



# Tipo di dato Stack: applicazioni

- Il termine stack viene usato in informatica in modo più specifico in diversi contesti:
  - lo stack è un elemento dell'architettura dei moderni processori, e fornisce il supporto fondamentale per l'**implementazione del concetto di subroutine** (vedi call stack, ricorsione)
  - le macchine virtuali di quasi tutti i linguaggi di programmazione ad alto livello usano uno **stack dei record di attivazione** per implementare il concetto di subroutine (generalmente, ma non necessariamente, basandosi sullo stack del processore)
  - la memoria degli automi a pila dell'informatica teorica è uno stack

# Tipo di dato Stack: esempi didattici

- Verificare il bilanciamento delle parentesi in espressioni e programmi

`abc{defg{ijk}{l{mn}}op}qr`     `(true)`

`abc{def}}{ghij{kl}m`     `(false)`

`abc{def}{ghij{kl}m`     `(false)`

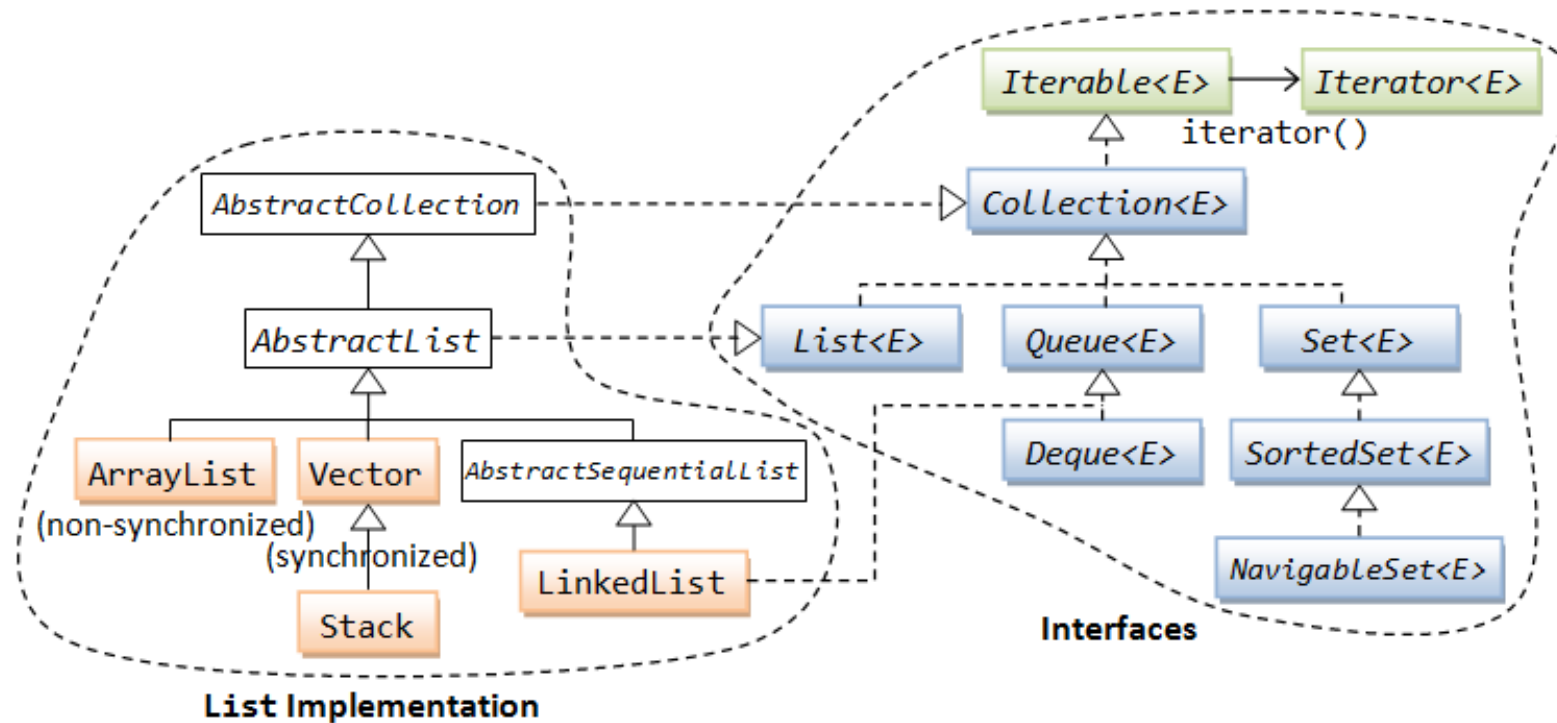
- Riconoscere stringhe palindrome

`abcdcba`

- Valutare espressioni postfisse

`2 3 4 + *`

# Stack in Java



- La classe **Stack** nel package `java.util` dovrebbe essere evitata poiché è una sottoclasse di **Vector** e perciò consente l'esecuzione di operazioni non-stack (**Rif. StackExample.java**)

# Vector<E> VS ArrayList<E> (cenno)

- Le classi generiche `Vector<E>` e `ArrayList<E>` sono sostanzialmente equivalenti, ma:
  - I metodi di `Vector<E>` sono **sincronizzati**, mentre quelli di `ArrayList<E>` non lo sono. Quindi se il programma è **concorrente** (cioè usa il **multi-threading** di Java) è opportuno usare `Vector<E>`, altrimenti conviene `ArrayList<E>` perché più efficiente.
  - `Vector<E>` fornisce, con opportuni metodi e costruttori, un controllo maggiore sulla **capacità**, cioè la dimensione dell'array sottostante.
  - Per motivi storici, `Vector<E>` fornisce più metodi con nomi diversi per manipolare gli elementi di un vettore.

# L'interfaccia MyStack

- La seguente interfaccia definisce le operazioni di interesse per uno stack (**Rif. MyStack.java**)

```
public interface MyStack<T> {  
    void push(T item);  
    T pop();  
    T peek();  
    int size();  
    boolean isEmpty();  
}
```

# Implementazioni

- Implementazione semplice basata su LinkedList: lo stack “delega” banalmente alla lista!

**Rif. `LinkedStack.java`**

- Implementazione semplice basata su ArrayList: lo stack “delega” banalmente alla lista!

**Rif. `ArrayListStack.java`**

- Implementazione basata su array (tecnica del raddoppiamento-dimezzamento)

**Rif. `ArrayStack.java`**



# Applicazione: la notazione postfissa

Nella notazione **postfissa** gli operatori seguono gli operandi su cui operano:

**Infissa:**  $(5 + (9 + 8) * 7)$

**Postfissa:**  $5\ 9\ 8\ +\ 7\ *\ +$

**Infissa:**  $((11 + 22) * 33)$

**Postfissa:**  $11\ 22\ +\ 33\ *$

# Notazione postfissa: regole di calcolo

- Quando si valuta un'espressione aritmetica in forma infissa bisogna tener conto dell'ordine di precedenza e dell'associatività degli operatori. Sono necessarie le parentesi per controllare l'ordine delle operazioni.
- Nella notazione postfissa non è necessario l'uso delle parentesi:
  1. si scandisce l'espressione da sinistra a destra fino a che si raggiunge il primo operatore;
  2. si applica l'operatore ai due operandi alla sua sinistra, si sostituisce il risultato nell'espressione al posto di operandi e operatore.

# Valutazione espressione postfissa: esempio

71 10 **100 5 -** \* + 21 -  
71 **10 95** \* + 21 -  
**71 950** + 21 -  
**1021 21 -**  
**1000**

- La **pila** è la struttura dati che con maggior naturalezza supporta la valutazione di espressioni algebriche in forma postfissa.
- Nella fase di valutazione, si usa una pila come struttura ausiliaria per conservare gli operandi.

# Notazione postfissa: algoritmo per la valutazione di espressioni algebriche

Scandisci l'espressione da sinistra a destra:

- se leggi un operando:
  - impila l'operando nella **pila degli operandi**
- se leggi un operatore (binario):
  - rimuovi dalla pila l'operando in cima e salvalo in **op1**
  - rimuovi dalla pila l'operando in cima e salvalo in **op2**
  - applica l'operatore a op2 e op1 e impila il risultato in cima alla pila.
- Al termine della scansione il risultato è nella pila



UNIVERSITÀ  
DEGLI STUDI  
DELL'AQUILA



DISIM  
Dipartimento di Ingegneria  
e Scienze dell'Informazione  
e Matematica



# Domande?

**Giovanna Melideo**  
Università degli Studi dell'Aquila  
DISIM