

PILE

CARATTERISTICHE GENERALI)

- Pila di oggetti che possono essere inseriti ed estratti secondo comportamento **LIFO**
 - **Last In, First Out**
 - possono essere inseriti/estratti/ispezionati solo dalla cima della pila

UTILIZZO DI PILE

- Browser
 - andare alla pagina precedente/successiva
- Editor di testi
 - Control Z (operazione di undo)
- Java Stack
 - utilizzata nella JVM

INTERFACCIA

- Definisce le operazioni
 - **push**: inserisce un oggetto in cima alla pila
 - **pop**: elimina l'oggetto che si trova in cima alla pila
 - **top**: ispeziona elemento in cima alla pila

```
public interface Stack extends Container {  
    void push(Object obj);  
    Object pop();  
    Object top();  
}
```

REALIZZAZIONE DELLA PILA

- Struttura dati: **array "riempito solo in parte"**
- Definiamo due nuove **eccezioni**
 - `class EmptyStackException extends RuntimeException`
 - `pop()` su array vuoto
 - `class FullStackException extends RuntimeException`
 - `push()` su array pieno

- Senza ridimensionamento:

```
1 public class FixedArrayStack implements Stack {
2     public final static int INIT_SIZE = 100;
3     protected Object[] array;
4     protected int arraySize = 0;
5
6     public FixedArrayStack() {
7         array = new Object[INIT_SIZE];
8         makeEmpty();
9     }
10
11     public boolean isEmpty() {
12         return arraySize == 0;
13     }
14
15     public void makeEmpty() {
16         arraySize = 0;
17     }
18
19     public void push(Object obj) {
20         if (arraySize == array.length)
21             throw new FullStackException();
22         array[arraySize++] = obj;
23     }
24
25     public Object top() {
26         if (isEmpty())
27             throw new EmptyStackException();
28         return array[arraySize - 1];
29     }
30
31     public Object pop() {
32         Object obj = top();
33         arraySize--;
34         return obj;
35     }
36 }
37
```

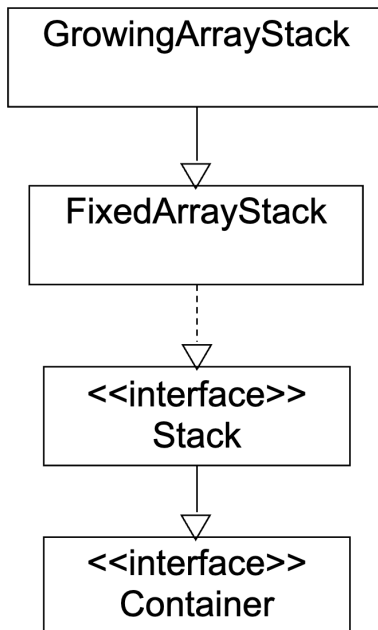
- Con Ridimensionamento

```

1  public class GrowingArrayStack extends FixedArrayStack {
2      public void push(Object obj) {
3          if (array.length == arraySize) {
4              array = resize(2 * arraySize);
5          }
6          array[arraySize++] = obj;
7      }
8
9      protected Object[] resize(int newLength) // solita tecnica
10     {
11         if (newLength < array.length)
12             throw new IllegalArgumentException();
13         Object[] newArray = new Object[newLength];
14         System.arraycopy(array, 0, newArray, 0, array.length);
15         return newArray;
16     }
17 }
18

```

- Nel complesso:



ANALISI PRESTAZIONI

- Dipendono dalla **definizione** della **struttura dati** e non dalla sua interfaccia
- FixedArrayStack
 - tempo esecuzione di ogni operazione costante: $O(1)$
- Tempo di esecuzione di GrowingArrayStack
 - unica differenza è `push`

ANALISI AMMORTIZZATA DELLE PRESTAZIONI ASINTOTICHE

- Si applica all'analisi di tempi di esecuzione dei **metodi di inserimento** in strutture dati
- Analisi del **tempo di esecuzione medio** nel caso **peggiore**
- su **push** con costante **moltiplicativa**
 - (n-1) volte senza resize: $O(1)$
 - n-esima volta: resize $O(n)$
 - $T(n) = \frac{[(n-1)*O(1)+O(n)]}{n} = \frac{O(n)}{n} = O(1)$
- su **push** con costante **addittiva**
 - dimensione diventa $n + k$
 - operazioni lente sono $\frac{n}{k}$ (ogni k elementi devo effettuare un resize) e sono $O(n)$
 - operazioni veloci senza resize sono dunque $n - \frac{n}{k}$
 - sia $n - \frac{n}{k} = \frac{k-1}{k}n = O(n)$ e $\frac{n}{k} = \frac{1}{k}n = O(n)$
 - $T(n) = \frac{(n-\frac{n}{k})*O(1)+(\frac{n}{k})*O(n)}{n} = \frac{O(n)+n*O(n)}{n} = \frac{O(n)}{n} + O(n) = O(1) + O(n) = O(n)$
- Considerazioni generali
 - push ha prestazioni $O(1)$ per qualsiasi costante **moltiplicativa**
 - push ha prestazioni $O(n)$ per qualsiasi costante **addittiva**

PILE DI DATI FONDAMENTALI

- Trasformare dato fondamentale in oggetto attraverso **classi involucro (wrapper)**

```
Integer myIntObj1 = new Integer(2);
Integer myIntObj2 = 2; // sintassi con auto-boxing
int myInt1 = myIntObj1.intValue();
int myInt2 = myIntObj2; // sintassi con auto-unboxing
```