ADT stack

Lo **stack** viene detto anche **pila** o **catasta** o ancora **coda LIFO** (Last In First Out) e possiamo figuracela proprio come un un mucchio di oggetti a cui possiamo aggiungere oggetti solo appoggiandoli sulla sommità e viceversa è possibile prendere solo il primo in alto. La conseguenza è che il primo oggetto disponibile è l'ultimo inserito, in altre parole l'ordine di estrazione è l'inverso dell'ordine di inserzione.

Lo stack è una struttura di dati molto utilizzata in ambito informatico, già abbiamo citato che ad ogni chiamata di esecuzione di metodo viene caricato in stack il record di attivazione con tutto il suo ambiente.

Proprietà di stack

Lo stack può essere vuoto, e quindi da uno stack vuoto non è possibile estrarre dati. All'estremo opposto potrà succedere che uno stack abbia saturato lo spazio di memoria disponibile e quindi non sarà possibile inserire ulteriori dati. Inoltre lo stack struttura dati tutti dello stesso tipo. Dopo questa premessa ecco le possibili operazioni di stack.

- Inserzione di un dato d. A differenza di quanto fatto per la sequenza non si indica il punto di inserzione perché come già detto il punto di inserzione deve essaere la cima dello stack. Questa operazione viene chiamata **push**.
- Elimina il dato in cima allo stack. Questa operazione chiamata **pop**.
- Get del valore in cima allo stack. Questa operazione viene chiamata **top**. In alcune implementazioni si può trovare una unica operazione che congiuntamente realizza top e pop, ma noi le terremo distinte.
- Test se lo stack è vuoto.

Definizione della interfaccia

Ecco la formalizzazione in java dell'elenco delle operazioni descritte:

```
/**
 * Proprietà di stack
 * @author giovanni.ragno
 */
public interface Stack<T> {
  /**
   * inserisce un dato in cima allo stack
   * @param dato da inserire
   */
  void push(T dato);
  /**
   * elimina il dato in cima allo stack
   */
  void pop();
  /**
   * restituisce il dato in cima allo stack
   * @return il dato
   */
```

```
  T top();
  /**
   * test se lo stack è vuoto
   * @return true se vuoto, false in caso contrario
   */
  boolean isEmpty();
}
```

Implementazione di stack

Come già fatto per la sequenza possiamo implementare uno stack basandoci su un array.

Possiamo però osservare anche che lo stack può essere considerato come una sequenza con particolari restrizioni d'uso.

Per cui definiremo due classi di implementazione di stack, la prima StackArray basata sull'uso di un array e la seconda StackSequenza basata sull'uso di una sequenza. Sarà interessante confrontare sie le prestazioni sia il tempo e sforzo necessario per lo sviluppo.