Liste, Stack e Alberi

Es#1: Lista di Interi

Si progetti ed implementi una lista di interi in java tramite rappresentazione concatenata. La lista è caratterizzato da tre operazioni:

`void` **`ins`** `(int num):` inserisce un intero in testa alla lista.
`void` **`rimuovi`** `(int num):` elimina l'elemento num specificato dalla lista. Genera una eccezione nel caso in cui la lista sia vuota o nel caso in cui l'elemento non sia presente nella lista.
`boolean` **`esiste`** `(int num):` restituisce true se l'elemento specificato è presente nella lista, false viceversa. Genera una eccezione nel caso in cui la lista sia vuota.

Es#2: Lista Ordinata Interi

Si progetti ed implementi una lista di interi in java tramite rappresentazione concatenata. La lista è caratterizzato da tre operazioni:

`void` **`ins`** `(int num):` inserisce un intero nella lista qualora non sia stato precedentemente inserito. Il numero deve essere inserito mantenendo l'ordinamento crescente della lista.
`void` **`rimuovi`** `(int num):` elimina l'elemento in testa alla lista. Genera una eccezione nel caso in cui la lista sia vuota o nel caso in cui l'elemento non sia presente nella lista.
`boolean` **`esiste`** `(int num):` restituisce true se l'elemento specificato è presente nella lista, false viceversa. Genera una eccezione nel caso in cui la lista sia vuota.

Es#3: Stack

Si richiede di sviluppare un componente software **Stack** che implementa i seguenti metodi.

- **push**: inserisce un elemento nello stack
- **pop**: preleva l'ultimo elemento inserito nello stack. Solleva una eccezione nel caso in cui lo stack sia vuoto.
- **top**: restituisce il valore dell'ultimo elemento inserito nello stack ma non lo elimina. Solleva una eccezione nel caso in cui lo stack sia vuoto.

Es#4: Alberi

Si richiede di progettare ed implementare un componenete che gestisce una struttura dati ad albero binario di ricerca per un insieme di oggetti generico. In particolare, gli oggetti devono implementare l'interfaccia comparable e la relazione d'ordine fra di essi deve essere imposta dal metodo compareTo. Il componente deve consentire di:

1) Inserire un nuovo oggetto nell'albero
2) Verificare se un oggetto è stato inserito in precedenza
3) Visitare completamente tramite visita in-order l'albero e stampare a video la stringa ottenuta invocando il metodo toString() dell'oggetto memorizzato in ogni nodo

Es#4: Alberi

Si richiede implementare una classe Studente che rappresenta gli studenti di un corso. Ogni studente è rappresentato da nome (stringa), cognome (stringa) e matricola (intero).
Ai fini dell'applicazione due istanze della classe studente sono equivalenti se hanno la stesso valore dell'attributo matricola. Inoltre, ai fini dell'applicazione una istanza della classe studente è < di un'altra se è < il valore della matricola.

Es#4: Alberi

Si richiede di implementare un componente Main che implementa il main della applicazione. Si richiede in particolare che:

1) Venga istanziato il componente che implementa l'albero
2) Vengano creati due studenti
    1) Homer Simpson avente matricola 11
    2) Bart Simpson avente matricola 12
3) Vengano inseriti gli studenti nell'albero
4) Venga stampato l'albero