Δομές Δεδομένων & Ανάλυση Αλγορίθμων
3ο Εξάμηνο

Υλοποίηση Στοίβας και Ουράς με Συνδεδεμένη Λίστα

Δημοσθένης Σταμάτης
http:/www.iee.ihu.gr/~demos
Τμήμα Μηχανικών Πληροφορικής & Ηλεκτρονικών Συστημάτων

Στοίβα (Stack)

☞ Στοίβα (stack) είναι μία λίστα στην οποία νέα στοιχεία μπορούν να προστεθούν και να αφαιρεθούν μόνο από τη μία άκρη της (κορυφή της στοίβας). Μία στοίβα αναφέρεται και σαν μία λίστα τύπου LIFO (Last-In-First-Out)

☞ Η Στοίβα σαν Αφηρημένος Τύπος Δεδομένων στη Java:

```
public interface Stack
{
public int size( );
  // επιστρέφει το μέγεθος της στοίβας
public boolean isEmpty( );
  // αληθεύει εάν η στοίβα είναι κενή
public Object top( ) throws StackEmptyException;
  // επιστρέφει το στοιχείο που βρίσκεται στην κορυφή της στοίβας
public void push(Object item) throws StackFullException;
  // εισάγει ένα νέο στοιχείο στην κορυφή της στοίβας
public Object pop( ) throws StackEmptyException;
  // εξάγει και επιστρέφει το στοιχείο που βρίσκεται στην κορυφή της στοίβας
}
```

```
public class LinkedStack implements Stack
// Υλοποίηση στοίβας με την τεχνική της σύνθεσης (composition)
// Η τεχνική αυτή ονομάζεται και Εξουσιοδότηση (Delegation)
 {
        private LinkedList S;

        public LinkedStack( ) {
          S=new LinkedList();
        }

        public int size() {
          return S.size();
        }

        public boolean isEmpty() {
           return S.isEmpty();
        }
```



```
// public class LinkedStack implements Stack (συνέχεια)
// Υλοποίηση στοίβας με την τεχνική της σύνθεσης (composition)
// Η τεχνική αυτή ονομάζεται και Εξουσιοδότηση (Delegation)

    public Object top( ) throws StackEmptyException{
        if (S.isEmpty()) throw new StackEmptyException("Empty Stack!");
        Object temp = S.removeFirst();
        S.insertFirst(temp);
        return temp;
        }
```

```
// public class LinkedStack implements Stack (συνέχεια)
// Υλοποίηση στοίβας με την τεχνική της σύνθεσης (composition)
// Η τεχνική αυτή ονομάζεται και Εξουσιοδότηση (Delegation)

        public void push(Object item) throws StackFullException {
          S.insertFirst(item);
        }

        public Object pop( ) throws StackEmptyException {
          try {
                return S.removeFirst();
          }
          catch (ListEmptyException str) {
                throw new StackEmptyException("EmptyStack");
          }
        }
```



```
import Node;
public class LinkedStack2 implements Stack
// Υλοποίηση της στοίβας με τη χρήση μόνον της κλάσης Node
{
        private Node top;
        private int size;

        public LinkedStack2( ) {
          top = null;
          size = 0;
        }

        public int size( ) {
          return size;
        }

        public boolean isEmpty( ) {
           return (top == null);
        }
```

```
// import Node;
// public class LinkedStack2 implements Stack (συνέχεια)
// Υλοποίηση της στοίβας με τη χρήση μόνον της κλάσης Node

        public Object top( ) throws StackEmptyException {
          if (isEmpty( )) throw new StackEmptyException("Empty Stack!");
          return top.getItem();
        }

        public void push(Object item) throws StackFullException {
          Node newTop = new Node( );
          newTop.setItem(item);
          newTop.setNext(top);
          top = newTop;
          size++;
        }
```



```
// import Node;
// public class LinkedStack2 implements Stack (συνέχεια)
// Υλοποίηση της στοίβας με τη χρήση μόνον της κλάσης Node

        public Object pop( ) throws StackEmptyException {
          try {
                Object temp = top.getItem();
                top = top.getNext();
                size--;
                return temp;
          }
          catch (NullPointerException str) {
                throw new StackEmptyException("EmptyStack");
          }
        }
}
```

```
public class LinkedStack3 extends LinkedList implements Stack

// Υλοποίηση της στοίβας με τη μέθοδο της κληρονομικότητας
// από την κλάση LinkedList

// Ερώτηση: Τί γίνεται με τις άλλες μεθόδους της LinkedList?

{
        public LinkedStack3( )
        { super( ); }

        public int size() {
          return super.size();
        }

        public boolean isEmpty() {
           return super.isEmpty();
        }
```



```
// public class LinkedStack3 extends LinkedList implements Stack (συνέχεια)
// Υλοποίηση της στοίβας με τη μέθοδο της κληρονομικότητας
// από την κλάση LinkedList

        public Object top( ) throws StackEmptyException{
          if (isEmpty( )) throw new StackEmptyException("Empty Stack!");
          Object temp;
          temp = removeFirst();
          insertFirst(temp);
          return temp;
        }

        public void push(Object item) throws StackFullException{
          insertFirst(Object);
        }
```

```
// public class LinkedStack3 extends LinkedList implements Stack
 (συνέχεια)
// Υλοποίηση της στοίβας με τη μέθοδο της κληρονομικότητας
// από την κλάση LinkedList


        public Object pop( ) throws StackEmptyException {
          try {
                return S.removeFirst();
              }
         catch (ListEmptyException str) {
                throw new StackEmptyException("EmptyStack");
              }
}
```



```
public class LinkedQueue implements Queue
// Υλοποίηση ουράς με την τεχνική της σύνθεσης (composition)
 {
        private LinkedList Q;

        public LinkedQueue( ) {
          Q=new LinkedList();
        }

        public int size() {
          return Q.size();
        }

        public boolean isEmpty() {
           return Q.isEmpty();
        }
```

```
// public class LinkedQueue implements Queue - συνέχεια
// Υλοποίηση ουράς με την τεχνική της σύνθεσης (composition)

  public Object front( ) throws QueueEmptyException
  {
      Object temp;
      if (Q.isEmpty()) throw new QueueEmptyException("Empty Queue!");
      temp = Q.removeFirst();
      Q.insertFirst(temp);
      return temp;
  }

  public void enqueue(Object item) throws QueueFullException
  {
      Q.insertLast(item);
  }
```



```
// public class LinkedQueue implements Queue - συνέχεια
// Υλοποίηση ουράς με την τεχνική της σύνθεσης (composition)


   public Object dequeue( ) throws QueueEmptyException
   {

          try {
               return Q.removeFirst();
                }
          catch (ListEmptyException str) {
               throw new QueueEmptyException("EmptyQueue");
              }
   }
```