/**
 * Solution to CharStack
 */

public class CharStack implements StackInter {
   // instance variable to mark top of stack
   Node top = null;

   // inner (aka nested) class Node
   private class Node {
      Character item;
      Node next;
   }

   /**
   * push takes a Character ch as input and makes it
   * the top of the stack. There are 2 cases, one
   * in which the stack is empty: if so, top
   * is instantiated as a new Node. Its item field is
   * set to ch and its next field to null. In the
   * other case, stack is non-empty, so a new
   * Node is created, its item field set to ch and
   * its next field to whatever the next field of
   * top is pointing to. In each case, set top to be
   * the new Node.
   */
   public void push(Character ch) {
      if (this.isEmpty()) { //stack is empty
         top = new Node();
         top.item = ch;
         top.next = null;
      } else { // stack is not empty
         Node adding = new Node();
         adding.item = ch;
         adding.next = top;
         top = adding;
      }
   }

   /**
   * pop displays an error message if the stack is empty.
   * Otherwise, the Node top is saved, top is set to top's
   * next field, and the Character that was stored in top
   * is returned.
   */
   public Character pop() {
      if (this.isEmpty())
         System.out.println("Stack is empty, pop impossible");
      Node removing = top;
      top = top.next;
      return removing.item;
   }

   /**
   * peek displays an error message if the stack is empty.
   * Otherwise, it returns the Character stored in the top
   * Node.
   */
   public Character peek() {
      if (this.isEmpty())
         System.out.println("Stack is empty, peek impossible");
      return top.item;
   }

   /**
   * If top is a null pointer, return true
   */
   public boolean isEmpty() {
      return top == null;
   }
}
public String toString() {
    String retStr = "";
    if (this.isEmpty())
        return "Empty";
    else {
        while (!this.isEmpty()) {
            Character printed = this.pop();
            retStr += printed.toString() + " ";
            System.out.println("Popped " + printed.toString() + " from stack");
        }
        return retStr;
    }
}

/*
 * Solution to Stack interface
 */

public interface StackInter {
    // remove and return the top char from the stack
    public Character pop();

    // put a new char on the top of the stack
    public void push(Character ch);

    // return true if the stack is empty
    public boolean isEmpty();

    // return the value contained in the top node of the stack
    public Character peek();
}