```java
/**
 *  -----------------------------------------------------------------------------------
 * @Fichier      : Stack.java
 * @Labo         : Laboratoire 7 : Tours de Hanoï
 * @Auteurs      : Slimani Walid & Baume Oscar
 * @Date         : 09.11.2022
 *
 * @Description  : Ce fichier définit la classe Stack. Cette classe permet de
 *                 modéliser la strucutre de donnée "Pile".
 * @Remarque     : La pile modélisée est en réalité une liste simplement chaînée
 * @Modification : / Aucune modification
 *  -----------------------------------------------------------------------------------
 **/

package util;

public class Stack<T> {
    // region ctor

    /**
     * Nom          : Stack
     * Description  : Permet de construire une Stack avec un nombre d'élément variable
     * @param values : Valeur(s) de(s) élément(s) à ajouter dans la pile
     * @return       : L'objet Stack construit par le constructeur
     **/
    public Stack(T... values) {
        for (T val : values) {
            push(val);
        }
    }
    // endregion

    // region paramètre
    private ElementStack<T> head;   // Représente le sommet de la pile
    // endregion

    // region method

    /**
     * Nom           : push
     * Description   : Permet d'ajouter un élément au sommet de la pile
     * @param  value : valeur de l'élément à ajouter au sommet de la pile
     * @return       : void
     **/
    public void push(T value) {
        if (head == null) {
            head = new ElementStack<>(value);
        } else {
            head = new ElementStack<>(value, head);
        }
    }

    /**
     * Nom           : pop
     * Description   : Permet d'enlever l'élément se situant au sommet de la pile
     * @return       : L'élément de type <T> qui était au sommet de la pile
     **/
    public T pop() {
        if (head == null)
            throw new RuntimeException("pop() : La stack est vide");

        ElementIterator it = new ElementIterator(head);
        T value = head.getValue();
        head = it.next();
        return value;
    }

    /**
     * Nom           : top
     * Description   : Retourne la valeur de l'élément se situant au sommet de la pile
     * @return       : La valeur de type <T> de l'élément qui est au sommet de la pile
     **/
    public T top() {
        if (head == null) {
            throw new RuntimeException("top() : La stack est vide");
        }
        return head.getValue();
```

```
    }

    /**
     * Nom           : toString
     * Description   : Permet d'afficher la pile
     * @return       : String représentant l'état de la pile
     **/
    public String toString() {
        String rt = "[ ";
        ElementIterator it = new ElementIterator(head);

        if (head != null) {
            rt += "<" + head.getValue() + "> ";
            while (it.hasNext()) {
                rt += "<" + it.next().getValue() + "> ";
            }

        }
        return rt + "]\n";
    }

    /**
     * Nom           : toArray
     * Description   : Permet de convertir une stack en un tableau "d'Object"
     * @return       : Le tableau "d'Object"
     **/
    public Object[] toArray() {
        Object out[] = new Object[0];
        int pos = 0;
        if (head != null) {
            ElementIterator it = new ElementIterator(head);
            pos++;
            while (it.hasNext()) {
                it.next();
                pos++;
            }
            it = new ElementIterator(head);
            out = new Object[pos];
            pos = 0;
            out[pos++] = head.getValue();
            while (it.hasNext()) {
                out[pos++] = it.next().getValue();
            }
        }
        return out;
    }

    /**
     * Nom         : getHead
     * Description : retourne le sommet de la pile
     * @return     : Un ElementStack qui est le sommet de la pile
     * Remarque    : /!\ cette fonction a été implémenté UNIQUEMENT pour pouvoir tester le bon
     fonctionnement
     *               de la classe Stack (voir le test "Test des itérateurs" dans la classe "TestStack")
     /!\
     **/
    public ElementStack<T> getHead() {
        if(head == null) throw new RuntimeException("getHead() : la stack est vide");
        return head;
    }
    // endregion
}
```