util\Stack.java

```java
/**
 * @author  Sebastian Diaz & Guillaume Dunant
 * Date    : 16.11.2023
 * Fichier: Stack.java
 */
package util;

import java.lang.reflect.Array;
import java.util.Arrays;

/**
 * Implémentation de la pile
 */
public class Stack<T> {

    Node<T> head;
    int size = 0;

    /**
     * Constructeur
     */
    public Stack(){
        head = null;
    }

    /**
     * Retourne un itérateur pour parcourir la stack
     * @return Iterator<T>
     */
    public Iterator<T> getIterator(){
        Iterator<T> ite = new Iterator<>(head);
        return ite;
    }

    /**
     * Ajoute une valeur sur la pile
     * @param value Valeur à ajouter
     */
    public void push(T value){
        if (head == null) {
            head = new Node<T>(value);
        }
        else{
            head = new Node<T>(value, head);
        }
        ++size;
    }

    /**
     * Retourne et retire la dernière valeur de la stack
     * @return La valeur retirée
     */
    public T pop(){

        if(head == null){
            return null;
```

```
57          }
58          Node<T> top = head;
59          head = head.getPrevious();
60          --size;
61          return top.getValue();
62      }
63
64      /**
65       * Retourne les valeurs de la stacks en String
66       */
67      public String toString(){
68          T[] values = getValues();
69
70          if (values == null) {
71              return "[]";
72          }
73          else{
74              return Arrays.toString(getValues());
75          }
76      }
77
78      /**
79       * Retourne un tableau contenant les valeurs de la stacks
80       * @return T[] ou null si la stack est vide
81       */
82      public T[] getValues(){
83
84          if(head == null){
85              return null;
86          }
87
88          T[] values = (T[]) Array.newInstance(head.getValue().getClass(), size);
89          int counter = 0;
90          Iterator<T> ite = getIterator();
91          while(ite.hasNext()){
92              values[counter++] = ite.next();
93          }
94          return values;
95      }
96
97      /**
98       * Retourne le nombre d'élément de la stack
99       * @return int
100      */
101     public int size(){
102         return size;
103     }
104
105     /**
106      * Retire tous les éléments de la stack
107      */
108     public void emptyStack(){
109         head = null;
110         size = 0;
111     }
112 }
113
```