```cpp
/**
 * @file -  StackADT.h
 *
 * Defines StackADT<T> as an abstract data type.
 * Does not specify the internal data structure that is necessary to use to hold the stack elements.
 * An implementor of this abstract stack data type is free to choose the internal data structure.
 * Specifies the operations or the functions that this abstract stack data type needs to support
 * in order to become a Last in First out (LIFO) container or list.
 * Does not specify the algorithms of these operations or functions.
 * An implementor of this abstract stack data type is free to choose the algorithms for the operations
 * that suit best with his/her choice of internal data structure of this abstract data type.
 *
 * @author - Humayun Kabir, Instructor, CSCI 161, VIU
 * @version - 0.0.1
 * @date - April 25, 2021
 *
 */




#ifndef __STACKADT_H_INCLUDED__
#define __STACKADT_H_INCLUDED__

#include <iostream>

template <typename T>
class StackADT {
public:
    /*
     * Default constructor
     */
    StackADT() {
        std::cout<<"StackADT::constructor......"<<std::endl;
    }

    /*
     * Destructor
     */
    virtual ~StackADT() {
        std::cout<<"StackADT::desctructor......"<<std::endl;
    }

    virtual void push(T element) = 0;
    //Puts the element at the top of the stack

    virtual T pop() = 0;
    //Removes the top element from the stack

    virtual T peek() = 0;
    //Gives the top element of the stack without removing it.

    virtual int getSize() = 0;
    //Gives the current size or the number of elements that have been pushed but not popped yet.

    virtual bool isEmpty() = 0;
    //Returns true if the the stack is empty.
};

#endif

```