Stacks and Lists

This exercice focuses on structural design pattern training

Preamble

According to **UML class diagram** shown in UML_Diagram.pdf, produce code tant allow the provided application (that creates a StackOfStringTester object, bound to a ArrayStack object, and calls testStack method) to behave as expected.

Additional information

The *Stack* interface defines a stack (LIFO) of elements of type E, with the following behaviour

- the **push** method adds an element on top of the stack.
- the **pop** method removes the element on top of the stack and returns its value
- the **peek** method returns the value of the element on the top of the stack, without removing it
- the **size** method returns the number of elements currently stacked

The *ArrayStack* class is a **basic implementation** of the priorly described interface using an array as internal storage. The capacity of the stack is supposed to be 10, and it is optimistically supposed that the stack will never be full.

The *testStack* method:

- logs (i.e. prints on standard output) the size of the stack
- pushes "a" string on top of the stack, and logs it is done
- logs the size of the stack
- pushes "b" string on top of the stack, and logs it is done
- logs the size of the stack
- peeks the top of the stack, and logs its value
- logs the size of the stack
- pops the top of the stack, and logs its value
- logs the size of the stack
- pops the top of the stack, and logs its value
- logs the size of the stack

ADAPTER pattern

Without modifying any of the already written code, except application's main, apply the **ADAPTER pattern** to seamlessly make *StackOfStringTester* use an *ArrayList* object.

Additional information

The *List* interface defines a list (random access) of elements of type E, with the following behaviour

- the add method inserts an element at a given position (positions start at 0)
- the remove method removes the element at a given position
- the get method returns the value of the element at a given position, without removing it
- the size method returns the number of elements currently contained by the list

The *ArrayList* class is a **basic implementation** using an array as internal storage. The capacity of the list is supposed to be 10, and it is optimistically supposed that the list will never be full.

PROXY Pattern

Without modifying any of the already written code, except application's main, apply the **PROXY pattern** in order to control access to the stack:

- no StackOfStringTester object can call peek method
- some StackOfStringTester can only call push method
- some StackOfStringTester can only call pop method