```java
package adt;

/**
 * ArrayStack.java A class that implements the ADT array by using an expandable
 * array.
 *
 * @author Chew Lip Sin
 * @param <T> The type of elements stored in the stack.
 */
public class ArrayStack<T> implements StackInterface<T> {

    private T[] array;
    private int topIndex; // index of top entry
    private static final int DEFAULT_CAPACITY = 5;

    /**
     * Creates an ArrayStack with default capacity.
     */
    public ArrayStack() {
        this(DEFAULT_CAPACITY);
    }

    /**
     * Creates an ArrayStack with the given initial capacity.
     *
     * @param initialCapacity The initial capacity of the stack.
     */
    public ArrayStack(int initialCapacity) {
        array = (T[]) new Object[initialCapacity];
        topIndex = -1;
    }

    /**
     * Adds a new entry to the top of the stack.
     *
     * @param newEntry The object to be added as a new entry.
     */
    @Override
    public void push(T newEntry) {
        topIndex++;

        if (topIndex < array.length) {
            array[topIndex] = newEntry;
        }
    }

    /**
     * Retrieves the top entry of the stack without removing it.
     *
     * @return The top entry. If the stack is empty, returns null.
     */
    @Override
    public T peek() {
        T top = null;

        if (!isEmpty()) {
            top = array[topIndex];
        }

        return top;
    }

    /**
     * Removes and returns the top entry from the stack.
     *
     * @return The top entry. If the stack is empty, returns null.
     */
    @Override
    public T pop() {
        T top = null;
        if (!isEmpty()) {
            top = array[topIndex];
            array[topIndex] = null;
            topIndex--;

        } // end if

        return top;
    }

    /**
     * Checks if the stack is empty.
     *
     * @return True if the stack is empty, false otherwise.
     */
    @Override
    public boolean isEmpty() {
        return topIndex < 0;
    }

    /**
     * Removes all entries from the stack.
     */
    @Override
    public void clear() {
        topIndex = -1;
    }

    /**
     * Gets the number of entries currently in the stack.
     *
     * @return The number of entries.
     */
    @Override
    public int size() {
        return topIndex;
    }
}
```