

```
import java.util.Arrays;

/**
 * {@code StackOfInteger} represented as an array.
 *
 * @correspondence <pre>
 * {@code reverse(this) = this.elements[0:this.size-1] and
 * |this| = this.size}
 * </pre>
 * @convention <pre>
 * {@code 0 <= this.size <= this.elements.length}
 * </pre>
 *
 * @author Paolo Bucci
 */
public final class StackOfIntegerOnArray implements StackOfInteger {

    /**
     * Default initial size of array representation.
     */
    private static final int DEFAULT_CAPACITY = 10;

    /**
     * Array used to store the stack entries.
     */
    private int[] elements;

    /**
     * Integer used to keep track of the stack length.
     */
    private int size = 0;

    /**
     * Ensures that the array can accommodate a new entry and increases the size
     * of the array if necessary.
     *
     * @alters this.elements
     * @ensures {@code this.elements.length > this.size}
     */
    private void ensureCapacity() {
        if (this.elements.length == this.size) {
            this.elements = Arrays.copyOf(this.elements, 2 * this.elements.length + 1);
        }
    }

    /**
     * Default constructor.
     *
     * @ensures {@code this = <>}
     */
    public StackOfIntegerOnArray() {
        this(StackOfIntegerOnArray.DEFAULT_CAPACITY);
    }

    /**
     * Constructor to specify underlying representation initial memory
     * allocation.
     *
     * @param initialCapacity
     *         the number of preallocated entries
     * @requires {@code initialCapacity > 0}
     * @ensures {@code this = <> and [underlying representation may
     *         preallocate memory for initialCapacity entries]}
     */
}
```

```
    */  
    public StackOfIntegerOnArray(int initialCapacity) {  
        this.elements = new int[initialCapacity];  
    }  
  
    @Override  
    public void push(int x) {  
        this.ensureCapacity();  
        this.elements[this.size] = x;  
        this.size++;  
    }  
  
    @Override  
    public int pop() {  
        this.size--;  
        return this.elements[this.size];  
    }  
  
    @Override  
    public int top() {  
        return this.elements[this.size - 1];  
    }  
  
    @Override  
    public int length() {  
        return this.size;  
    }  
}
```