```java
/*
 * Copyright (c) 1994, 2010, Oracle and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 only, as
 * published by the Free Software Foundation.  Oracle designates this
 * particular file as subject to the "Classpath" exception as provided
 * by Oracle in the LICENSE file that accompanied this code.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Please contact Oracle, 500 Oracle Parkway, Redwood Shores, CA 94065 USA
 * or visit www.oracle.com if you need additional information or have any
 * questions.
 */

package java.util;

/**
 * The <code>Stack</code> class represents a last-in-first-out
 * (LIFO) stack of objects. It extends class <tt>Vector</tt> with five
 * operations that allow a vector to be treated as a stack. The usual
 * <tt>push</tt> and <tt>pop</tt> operations are provided, as well as a
 * method to <tt>peek</tt> at the top item on the stack, a method to test
 * for whether the stack is <tt>empty</tt>, and a method to <tt>search</tt>
 * the stack for an item and discover how far it is from the top.
 * <p>
 * When a stack is first created, it contains no items.
 *
 * <p>A more complete and consistent set of LIFO stack operations is
 * provided by the {@link Deque} interface and its implementations, which
 * should be used in preference to this class.  For example:
 * <pre>   {@code
 *   Deque<Integer> stack = new ArrayDeque<Integer>();}</pre>
 *
 * @author  Jonathan Payne
 * @since   JDK1.0
 */
public
class Stack<E> extends Vector<E> {
    /**
     * Creates an empty Stack.
     */
    public Stack() {
    }

    /**
     * Pushes an item onto the top of this stack. This has exactly
     * the same effect as:
     * <blockquote><pre>
     * addElement(item)</pre></blockquote>
     *
     * @param   item   the item to be pushed onto this stack.
     * @return  the <code>item</code> argument.
     * @see     java.util.Vector#addElement
     */
    public E push(E item) {
        addElement(item);

        return item;
    }

    /**
     * Removes the object at the top of this stack and returns that
     * object as the value of this function.
     *
     * @return  The object at the top of this stack (the last item
     *          of the <tt>Vector</tt> object).
     * @throws  EmptyStackException  if this stack is empty.
     */
    public synchronized E pop() {
        E       obj;
        int     len = size();

        obj = peek();
        removeElementAt(len - 1);

        return obj;
    }

    /**
     * Looks at the object at the top of this stack without removing it
     * from the stack.
     *
     * @return  the object at the top of this stack (the last item
     *          of the <tt>Vector</tt> object).
     * @throws  EmptyStackException  if this stack is empty.
     */
    public synchronized E peek() {
        int     len = size();

        if (len == 0)
            throw new EmptyStackException();
        return elementAt(len - 1);
    }

    /**
     * Tests if this stack is empty.
     *
     * @return  <code>true</code> if and only if this stack contains
     *          no items; <code>false</code> otherwise.
     */
    public boolean empty() {
        return size() == 0;
    }

    /**
     * Returns the 1-based position where an object is on this stack.
     * If the object <tt>o</tt> occurs as an item in this stack, this
     * method returns the distance from the top of the stack of the
     * occurrence nearest the top of the stack; the topmost item on the
     * stack is considered to be at distance <tt>1</tt>. The <tt>equals</tt>
     * method is used to compare <tt>o</tt> to the
     * items in this stack.
     *
     * @param   o   the desired object.
     * @return  the 1-based position from the top of the stack where
     *          the object is located; the return value <code>-1</code>
     *          indicates that the object is not on the stack.
     */
    public synchronized int search(Object o) {
        int i = lastIndexOf(o);

        if (i >= 0) {
            return size() - i;
        }
        return -1;
    }

    /** use serialVersionUID from JDK 1.0.2 for interoperability */
    private static final long serialVersionUID = 1224463164541339165L;
}

```