```
 1 package DataStructures;
 2
 3 import Exceptions.*;
 4
 5 // StackAr class
 6 //
 7 // CONSTRUCTION: with no initializer
 8 //
 9 // ******************PUBLIC OPERATIONS*********************
10 // void push( x )       --> Insert x
11 // void pop( )          --> Remove most recently inserted item
12 // Object top( )        --> Return most recently inserted item
13 // Object topAndPop( )  --> Return and remove most recent item
14 // boolean isEmpty( )   --> Return true if empty; else false
15 // void makeEmpty( )    --> Remove all items
16 // ******************ERRORS********************************
17 // top, pop, or topAndPop on empty stack
18
19 /**
20  * Array-based implementation of the stack.
21  */
22 public class StackAr implements Stack
23 {
24     public StackAr( )
25       { /* Figure 15.3 */ }
26
27     public boolean isEmpty( )
28       { return topOfStack == -1; }
29     public void makeEmpty( )
30       { topOfStack = -1; }
31     public void push( Object x )
32       { /* Figure 15.4 */ }
33     public Object top( ) throws Underflow
34       { /* Figure 15.5 */ }
35     public void pop( ) throws Underflow
36       { /* Figure 15.5 */ }
37     public Object topAndPop( ) throws Underflow
38       { /* Figure 15.6 */ }
39
40     private Object [ ] theArray;
41     private int        topOfStack;
42
43     static final int DEFAULT_CAPACITY = 10;
44
45     private void doubleArray( )
46       { /* Usual stuff, not shown */ }
47 }
```

Figure 15.2 `StackAr` class skeleton